The Diffusive Finite State Projection Algorithm for Efficient Simulation of the Stochastic Reaction-Diffusion Master Equation

Brian Drawert[15], Michael J Lawson[25], Linda Petzold[3], Mustafa Khammash[4]

January 4, 2010

Abstract

We have developed a computational framework for accurate and efficient simulation of stochastic spatially inhomogeneous biochemical systems. The new computational method employs a fractional step hybrid strategy. A novel formulation of the Finite State Projection (FSP) method, called the Diffusive FSP (DFSP) method, is introduced for the efficient and accurate simulation of diffusive transport. Reactions are handled by the Stochastic Simulation Algorithm (SSA).

Keywords: Spatial Stochastic Simulation, Reaction-Diffusion Master Equation, Finite State Projection

1 Introduction

On the cellular level of biological systems, molecules with small copy number interact randomly. The resulting fluctuations, or noise, in cellular species, play an important role in cell-cell variability and cell fate decisions. A classic example is the case of gene regulatory networks where low counts of genes and mRNA create stochastic effects that result in phenotypic differentiation [1][2]. Much recent work has focused on the development of efficient computational methods for discrete stochastic simulation of well-mixed biochemical systems [3]. However, the cell is *not* a spatially homogeneous environment. Spatial localization plays an important role in many cellular processes. For example, in the MinCDE

[1] *bdrawert@cs.ucsb.edu*, Dept. of Computer Science, University of California - Santa Barbara
[2] *mlawson@lifesci.ucsb.edu*, Dept. of Bio-Molecular Science & Engineering, University of California - Santa Barbara
[3] *petzold@cs.ucsb.edu*, Dept. of Computer Science, University of California - Santa Barbara
[4] *khammash@engineering.ucsb.edu*, Dept. of Mechanical Engineering, University of California - Santa Barbara
[5] These authors contributed equally to this work.

system of *Escherichia coli*, stochastic chemical reactions of spatially inhomogeneous species cause end-to-end oscillations [4]. In discussing the modeling of mutant phenotypes for this system, Feng and Elf [5] highlight the need for spatial stochastic simulations by noting that their "results emphasize that local copy number fluctuation may result in phenotypic differences although the total number of molecules of the relevant species is high." Additional examples are found in [6] and [7], among others.

Spatial stochastic simulation is an extremely computationally intensive task. This is due to the large number of molecules which, along with the refinement of the discretized spatial domain, results in a large number of diffusive transfers between sub-volumes. In this paper, we present a novel formulation of the Finite State Projection (FSP) method [8], called the Diffusive FSP (DFSP) method, for the efficient and accurate simulation of diffusive processes. Using the FSP method's ability to provide a bound on the error, we are able to take large diffusion timesteps with confidence in our solution. We then show how to construct a fractional step method for spatial stochastic simulation of reaction-diffusion processes which treats diffusion with DFSP and reactions with SSA.

The dynamics of spatially inhomogeneous stochastic systems are governed by the Reaction-Diffusion Master Equation (RDME) which was originally proposed and derived in [9]. More recently it was shown that biologically observed self-organized criticality emerges only when diffusion and reactions are treated as discrete stochastic processes [10]. This led to the adaptation of Gillespie's SSA to spatially inhomogeneous problems, called the Inhomogeneous SSA, or ISSA. In this formulation, the domain is discretized into subvolumes or voxels. Each voxel is well-mixed so that intra-voxel reactions are unchanged from the homogeneous case. Diffusive transfers between voxels are modeled by unimolecular decay and creation events occurring simultaneously in adjacent voxels. The state of the system is then the number of molecules of each species in each voxel at a given time.

It is important to note that the spatially inhomogeneous stochastic model is formulated on the mesoscopic scale. The voxel size is bounded by the well-mixed assumption of its mathematical formulation. We need to choose the length ℓ of a voxel small enough to capture the desired features of our system, but large enough so that the system can be considered to be well mixed in each voxel. Specifically, ℓ should satisfy $\ell \gg Kn/D$, where K is the reaction rate constant, n is the number of

molecules in a given voxel and $D = D_A + D_B$ is the combined diffusion rates of the reactants [11]. It is possible to reduce the voxel size by correcting the reaction propensities, down to a hard limit of $\ell \geq \beta_\infty K/D$, where $\beta_\infty \approx 0.25272$ [12].

Recent efforts have focused on speeding up the ISSA. The Next Subvolume Method (NSM) [7] is an efficient formulation of the ISSA for reaction-diffusion systems. NSM utilizes the priority queue structure found originally in the Next Reaction Method [13]. MesoRD [14] is a widely used implementation of this algorithm. The binomial tau-leap spatial stochastic simulation algorithm [15] seeks to improve performance by combining the ideas of aggregating diffusive transfers with the priority queue structure found in the NSM. The Multinomial Simulation Algorithm (MSA)[16] employs another strategy to improve performance. Noting that fast diffusive transfers between voxels often dominate the computational cost, MSA aggregates the diffusive transfers. Instead of executing each diffusive event individually, it calculates the inter-voxel flux of particles by sampling from a binomial distribution.

Under some circumstances it is possible to treat diffusion deterministically, thus eliminating the tracking of fast diffusive transfers almost entirely. Reactions are typically handled by the SSA. The Hybrid Multiscale Kinetic Monte Carlo Method [17] and the Gillespie multi-particle method [18] are examples of this approach. The adaptive hybrid method for stochastic reaction-diffusion processes described in [19] and implemented as part of the URDME software [20] integrates multiple methods for stochastic and deterministic diffusion adaptively for different components of a model.

The remainder of this paper is organized as follows. The next section briefly reviews the mathematical background, including the Chemical Master Equation (CME), SSA, FSP, RDME and ISSA on which our method is built. Section three describes the DFSP method and shows how to combine it with SSA or tau-leaping for reaction events to solve reaction-diffusion problems. In section four we present numerical experiments that demonstrate the speed and reliability of the new computational method. Finally, we conclude with an assessment of the proposed DFSP method, possible applications and future directions.

2 Background

In this section we briefly review the CME, SSA, and FSP algorithms for well-mixed chemical reacting systems, as well as the RDME and ISSA algorithm for spatially inhomogeneous systems.

2.1 CME and SSA

Consider a system involving N molecular species $\{S_1, ..., S_N\}$, represented by the state vector $X(t) = [X_1(t), ..., X_N(t)]^T$, where $X_i(t)$ is the number of molecules of species S_i at time t. There are M reaction channels, labeled $\{R_1, ..., R_M\}$, in the system. Assume the system is well-mixed and in thermal equilibrium. The dynamics of reaction channel R_j are characterized by the *propensity function* a_j and by the *state change vector* $\nu_j = [\nu_{1j}, ..., \nu_{Nj}]^T$: $a_j(x)dt$ gives the probability that, given $X(t) = x$, one R_j reaction will occur in the next infinitesimal time interval $[t, t + dt]$, and ν_{ij} gives the change in X_i induced by one R_j reaction.

The system is a Markov process whose dynamics are described by the Chemical Master Equation (CME) [21]

$$\begin{aligned}\frac{\partial P\left(x,t|x_0,t_0\right)}{\partial t} &= \mathcal{M}\, P\left(x,t|x_0,t_0\right) \\ &= \sum_{j=1}^{M}\left[a_j(x-\nu_j)P(x-\nu_j,t|x_0,t_0) - a_j(x)P(x,t|x_0,t_0)\right], \end{aligned} \tag{1}$$

where the function $P(x, t|x_0, t_0)$ denotes the probability that $X(t)$ will be x, given that $X(t_0) = x_0$ and $\mathcal{M}$ denotes the generating matrix for the Markov chain that describes the chemical reactions. For all but the most simple systems, the chemical master equation is made up of an extremely large or infinite number (dimension) of coupled ordinary differential equations (ODEs). Rather than evolve the CME directly, it is common practice to compute an ensemble of stochastic realizations whose probability density function converges to the solution of the CME. In chemical kinetics, the SSA [22] is used for this purpose.

At each step, the SSA generates two random numbers, r_1 and r_2 in $U(0, 1)$ (the set of uniformly distributed random numbers in the interval (0,1)). The time for the next reaction to occur is given

by $t + \tau$, where τ is given by

$$\tau = \frac{1}{a_0} \ln \left(\frac{1}{r_1} \right). \tag{2}$$

The index μ of the occurring reaction is given by the smallest integer satisfying

$$\sum_{j=1}^{\mu} a_j > r_2 a_0, \tag{3}$$

where $a_0(x) = \sum_{j=1}^{M} a_j(x)$. The system states are updated by $X(t + \tau) = X(t) + \nu_\mu$. The simulation then proceeds to the time of the next reaction. Because the SSA simulates all reaction events in the system, it can be computationally intensive. Much recent effort has gone into speeding up the SSA by reformulation [23], [13], [24], use of advanced computer architecture [25], and by aggregating reaction events to take larger time steps (tau-leaping)[26].

2.2 FSP

The Finite State Projection FSP [8] method directly calculates an analytical approximation to the solution of the CME, as opposed to simulating an ensemble of trajectories by SSA. It does this by forming a computationally tractable projection of the full state space and computing the time evolution of the probability density function in this projection space. The FSP was formulated to solve spatially homogenous stochastic models, but can be adapted to solve the diffusion master equation (DME). Techniques for taking advantage of time scale separation in spatially homogenous chemically reaction system were explored in [27] and [28].

The FSP method determines the approximate probability density vector (PDV) of the populations in a chemically reacting system by solving the CME in a truncated state space. Two theorems provide the foundation for the FSP. The first shows that the solution of the projected system increases monotonically as the size of the projection increases. The second guarantees that the approximate solution never exceeds the actual solution, and provides a bound on the error. It is important to note that while the evolution of a trajectory is random, the evolution of the PDV for a given initial condition is deterministic.

For a truncated state transition matrix A_J (see [8] for its construction) and initial truncated PDV $P_J(t=0)$, the FSP finds $P_J(t)$ at any time t within any given accuracy ϵ using the truncated CME

$$\dot{P}_J = A_J\, P_J(t). \tag{4}$$

Since (4) is a linear constant-coefficient ODE, its solution is given by

$$P_J(t) = \exp(A_J t) P_J(0). \tag{5}$$

Recent work has focused on optimizing FSP through more effective dynamic state space truncation [29] and more efficient algorithms for solving the resulting equation [30].

2.3 RDME and ISSA

Assume now that the domain Ω in space is partitioned into voxels V_k, $k = 1, ..., K$. For simplicity of presentation, we will assume for the moment that the domain is in one dimension. Each molecular species in the domain is represented by the state vector $X_i(t) = [X_{i,1}(t), ..., X_{i,K}(t)]$, where $X_{i,k}(t)$ is the number of molecules of species S_i in voxel V_k at time t. Molecules in the domain are able to react with molecules within their voxel, as described in section 2.1, and diffuse between neighboring voxels. The dynamics of diffusion of species S_i from voxel V_k to V_j is characterized by the *diffusion propensity function* $d_{i,k,j}$ and the *state change vector* $\mu_{k,j}$. $\mu_{k,j}$ is a vector of length K with -1 in the kth position, 1 in the jth position and 0 everywhere else: $d_{i,k,j}(x)dt$ gives the probability that, given $X_{i,k}(t) = x$, one molecule of S_i will diffuse from voxel V_k to V_j in the next infinitesimal time interval $[t, t + dt]$. Note that if $k = j \pm 1$ then $d_{i,j,k}(x) = D/l^2$, where D is the diffusion rate and l is the characteristic length of the voxel, and otherwise it is zero. The Diffusion Master Equation (DME)

can then be written in a form similar to the CME:

$$\begin{aligned}\frac{\partial P\left(x,t|x_0,t_0\right)}{\partial t} &= \mathcal{D}\,P\left(x,t|x_0,t_0\right)\\ &= \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{K}\left[d_{i,j,k}(x_i-\mu_{k,j})P(x_1,...,x_i-\mu_{k,j},...,x_N,t|x_0,t_0)-d_{i,j,k}(x_i)P(x,t|x_0,t_0)\right]\end{aligned} \tag{6}$$

where $\mathcal{D}$ denotes generating matrix for the Markov chain that describes the diffusion of molecules in the system.

The usual method of solution of the DME is to simulate each diffusive jump event explicitly, giving an exact solution. This is the method used by the ISSA and the NSM [7] algorithms. Another possibility is to use an approximate method to calculate the net inter-voxel transfers due to diffusion. The MSA [16] does this by realizing that the number of diffusion events conforms to a multinomial distribution which can be calculated and then sampled. The binomial tau-leap spatial stochastic simulation algorithm [15] uses a similar technique. In Section 3 we present a novel formulation of FSP that is used to find approximate solutions to the DME.

Combining (1) and (6) yields the RDME

$$\frac{\partial P\left(x,t|x_0,t_0\right)}{\partial t} = \mathcal{M}P\left(x,t|x_0,t_0\right)+\mathcal{D}P\left(x,t|x_0,t_0\right). \tag{7}$$

The RDME is a linear constant-coefficient ODE, however it has many more possible states than the corresponding CME and thus is more difficult to solve. Rather than solve the RDME directly, it is common practice to compute an ensemble of stochastic realizations whose histogram converges to the PDV of the RDME.

Many of the techniques for accelerating the SSA can be applied to the ISSA; however, the ISSA remains computationally expensive. The problem is that fast diffusive transfers between adjacent voxels dominate the computation time and limit the possibility for exploiting parallelism.

3 The Diffusive FSP Method

The DFSP method is based on two observations. First, diffusion of any one molecule is independent of the diffusion of all other molecules in the system. Using this independence, we note that the diffusion of molecules originating in one voxel is independent of the diffusion of all molecules originating in other voxels. Thus, we can decompose the problem of diffusing molecules in K voxels into K sub-problems, one for each voxel.

The second observation is that the DME describes a stochastic process, but the DME itself is an ODE and thus, deterministic. That is, the evolution of a particular trajectory is stochastic, but the evolution of the PDV describing the ensemble of many trajectories is deterministic. Thus, if we can solve the DME for a given sub-problem with n molecules for a time step Δt, then we can re-use this solution for any other sub-problem with n molecules and time step Δt. Next we will describe more rigorously a sub-problem and show how to set up and solve a FSP for such a sub-problem. Note that to solve the full problem, one needs only to sum the molecule distributions from each sub-problem.

3.1 DFSP

As above, we will consider a problem on a 1D periodic domain that is sub-divided into K equally sized voxels, each with length l. The kth sub-problem defines a diffusion problem that is initialized with empty voxels, except for the kth voxel, which contains n_k molecules of a given species. This initial condition is considered a state. The states of the system are defined by unique configurations of molecules in voxels, with the total number of molecules in the system always summing to n_k. The possible number of states is finite, though extremely large. The PDV enumerates these states and gives the probability of being in any state at a given time. For the initial condition, it is clear that the PDV for the system is $P(0) = [1, 0, 0...0]^T$. That is, at time zero, the probability of being in the state of the initial condition is one, and the probability of being in all other states is zero.

To solve the DME directly for a sub-problem, the DFSP method retains a finite set of states that carry a high probability and truncates states of little probabilistic importance. To determine which states to retain, we will walk through the process of diffusing molecules. The initial condition

forms the first tier. The second tier is defined by the states that can be reached with one diffusion event from the initial condition. The third tier is defined by the states that can be reached with one diffusion event from any state in the second tier and is not redundant with states in higher tiers.

In defining each of these states, there is an additional parameter, MAX, that is defined as the maximum number of voxels a particle can diffuse away from its originating voxel in one time step. The value of MAX is one less than the number of tiers. All of the states in the last tier are one diffusive step away from violating the MAX condition. MAX puts a limit on the allowable number of particles for a sub-problem without violating the error condition, (error $< \epsilon$). It is important to note that MAX dictates the amount of memory storage required by the algorithm.

For illustration, consider the situation where a voxel contains 20 molecules at the beginning of a time step, and $MAX = 2$; that is, we are tracking diffusive jumps of at most 2 voxels away from the originating voxel per time step. The initial state is given by $x_1 = \{0, 0, 20, 0, 0\}$. x_1 is the only state in the first tier. Since we are on a one-dimensional domain, the states reachable in a single diffusive jump event from x_1 are $x_2 = \{0, 1, 19, 0, 0\}$ and $x_3 = \{0, 0, 19, 1, 0\}$. These two states make up the second tier. The third tier is comprised of $x_4 = \{1, 0, 19, 0, 0\}$, $x_5 = \{0, 0, 19, 0, 1\}$, $x_6 = \{0, 2, 18, 0, 0\}$, $x_7 = \{0, 0, 18, 2, 0\}$ and $x_8 = \{0, 1, 18, 1, 0\}$. Note that x_1 is reachable from the states in the second tier, but since that state is found in a higher tier, it is not included in the third tier.

As each tier is added, the corresponding state transitions are included in A_J. After each tier is added, the truncated system can be solved and the truncated PDV ($P_J(\Delta t)$) calculated. Thus, after adding a tier, we can determine a bound on our error for the current projection (ϵ). The addition of states ends when the error bound is below a predetermined tolerance.

To calculate the final state of the system due to diffusion over an interval of Δt we sample the PDV by selecting K uniformly distributed random numbers $R_k \in U(0, 1)$ and finding the smallest integer μ_k such that $\sum_{j=1}^{\mu_k} PDV[j] > R_k$, where $PDV[j]$ is the probability weight of state j. Let $X_{s,k}(t) = n_k$ be the number of molecules of species s in voxel k at time t and let $T_\kappa(j \,|\, n)$ be the number of molecules in voxel κ of state x_j, given n molecules initially (e.g. $x_1 = \{0, 0, n, 0, 0\}$ if

MAX=2). Then the discrete time evolution of the system is given by

$$X_{s,k}(t+\Delta t) = \sum_{i=-MAX}^{MAX} T_i\left(\mu_{k+i} \,|\, X_{s,k+i}(t)\right). \tag{8}$$

For a sub-problem with n molecules and a time step Δt, we can store its PDV and re-use it for all other sub-problems containing n molecules and time step Δt. As a result, if we keep a constant time step, simulating a diffusion process becomes a matter of selecting K random numbers and performing a look-up and comparison.

To simulate the full RDME, we take a reaction step and then a diffusion step, each of size τ_D. Following the SSA, we take a reaction step by evolving the system through reaction events until the time of the next reaction exceeds τ_D. We then perform diffusion of the molecules at the end of the reaction step via the DFSP as described above. At the end of the diffusive step, the simulation time is $t_0 + \tau_D$. We continue interleaving reaction and diffusion steps until the final time.

3.2 Adaptive Step Splitting

In the case where an initial population for a sub-problem is large enough to exceed the error condition (ϵ) for a given MAX, we need to split the step. Rather than split the step in time, we take advantage of the independence of diffusing molecules and split the sub-problem into several sub-sub-problems. For example, suppose that the maximum number of particles one can diffuse in τ_D without violating the error condition is 10. In this case, we would treat this sub-problem of 20 particles as two sub-sub-problems of 10 each. The states for each sub-sub-problem are $x_1 = \{0, 0, 10, 0, 0\}$, $x_2 = \{0, 1, 9, 0, 0\}$, $x_3 = \{0, 0, 9, 1, 0\}$, $x_4 = \{1, 0, 9, 0, 0\}$, $x_5 = \{0, 0, 9, 0, 1\}$, $x_6 = \{0, 2, 8, 0, 0\}$, $x_7 = \{0, 0, 8, 2, 0\}$ and $x_8 = \{0, 1, 8, 1, 0\}$. We then can reconstruct the solution for the sub-problem by picking a uniformly distributed random number (as above) for each sub-sub-problem, selecting the corresponding state and then summing these two sets. It is clear that the states of the sub-problem are all possible combinations of x_1, x_2, x_3, x_4, x_5, x_6 and x_7. While some of these combinations may be redundant, the number of unique states for the sub-problem of 20 particles has been increased from the original 7. By the first FSP theorem, the solution of the projected system increases monotonically as the size

of the projection increases; as a corollary, the size of the error must decrease as we add states.

We continue splitting the sub-problems until the error from each sub-problem is less than $\epsilon/2^{\mathcal{L}}$, where $\mathcal{L}$ is the recursion level. In the extreme case of splitting the sub-problem into sub-sub-problems of one molecule, the combination of states would provide all possible combinations of the original n_k particles in the $2 \times MAX + 1$ voxels of the sub-problem.

The advantage of splitting the sub-problems in this way (as opposed to splitting the time step) is that we can keep Δt constant, which allows us to re-use our lookup table. Calculation of the lookup table is the most computationally expensive part of the algorithm. In order to maximize speed, we seek to avoid changing the time stepsize whenever possible.

Next we perform an analysis of the adaptive step splitting error control. Consider the case where we want to calculate a final state of a sub-problem containing 100 molecules of a chemical species after $\tau_D = 0.1$ using a local error tolerance of 10^{-5}. If all 100 molecules are moved simultaneously, then the resulting single step FSP error will be 0.38 and our truncated state space contains 62% of the probability density. Utilizing the fact that the FSP error has a non-linear relationship with the number of molecules moved (Figure 1 shows the error as a function of the number of molecules moved in one timestep), we can split the molecules into smaller groups where the sum of the error of diffusing the smaller groups is less than the original error. We recursively split a group of molecules in half if the error to move it in one step is greater than the error tolerance (adjusted for the recursion level). For 100 molecules, we first split them into two groups of 50 (error of $3.86e-2$), then four groups of 25 (error of $1.46e-3$), and so on. In total, we will move twelve groups of six molecules each with error $2.4e-7 < 10^{-5}/2^4 = 6.3e-7$ (four levels of recursion), four groups of four molecules each with error $1.4e-8 < 10^{-5}/2^5 = 3.1e-7$, and four groups of three molecules each with error $1.6e-9 < 10^{-5}/2^5 = 3.1e-7$ (both with five levels of recursion). The total error is $3.0e-6$ which is the sum of the error in all of the recursion steps. Using this method, we are able to satisfy the error tolerance, while continuing to utilize the efficiency of the lookup tables.

[Figure 1 here]

3.3 Detailed Algorithm Descriptions

State Space Exploration The algorithm to determine the truncated state space is presented in detail in Algorithm 1. The input parameters are the number n_k of particles in the originating voxel k, and the maximum number of diffusive transfers MAX that a particle can move away from the originating voxel in one diffusion time step. The state representing the initial condition is that all n_k particles are in the originating voxel. The algorithm is presented for an anisotropic, one dimensional Cartesian mesh with periodic boundary conditions, and assumes that the number of voxels in any dimension is large relative to MAX.

Algorithm 1 State Space Exploration

INPUT: n_k, MAX, Initial State
OUTPUT: $\texttt{TransitionMatrix}_n$, $\texttt{StateList}_n$

initialize: $\texttt{NextTierQueue} \leftarrow$ Initial State, $\texttt{Queue} \leftarrow \emptyset$
initialize: $\texttt{StateList} \leftarrow$ Initial State, $\texttt{TransitionMatrix} \leftarrow \emptyset$
for Tier $\in$ (2, MAX) **do**
  $\texttt{Queue} \leftarrow \texttt{NextTierQueue}$
  $\texttt{NextTierQueue} \leftarrow \emptyset$
  for all states $s \in \texttt{Queue}$ **do**
    for all non-empty voxels $v \in s$ **do**
      for all inter-voxel transitions d {with probabilities $p(d)$} originating from v **do**
        find state $t \leftarrow s + d(v)$
        if $t \notin \texttt{StateList}$ **then**
          add t to $\texttt{StateList}$, add t to $\texttt{NextTierQueue}$
        end if
        $\texttt{TransitionMatrix}(s,t) \leftarrow p(d)$
      end for
    end for
  end for
end for
Update Diagonal elements in $\texttt{TransitionMatrix}$
Truncate $\texttt{TransitionMatrix}$ so that it is of dimension $|\texttt{StateList}|$
Create absorbing state in $\texttt{TransitionMatrix}$

We then store the $\texttt{TransitionMatrix}_n$ and $\texttt{StateList}_n$ for later use. For all cases where the number of particles in the originating voxel n, such that n is greater than MAX, the structure of the $\texttt{TransitionMatrix}_n$ is constant, and the values in the matrix are linear functions of n. This matrix is obtained by performing the state space exploration algorithm with n as an unspecified parameter

constrained to a value greater than MAX. For $n < MAX$ it is still necessary to go through the state space exploration, because for these values the $\texttt{TransitionMatrix}_n$ will not conform to the general structure.

DFSP diffusion step This is the algorithm for taking a single time step of length τ_D for a single voxel k containing n_k particles. We assume that $\texttt{TransitionMatrix}_n$ and $\texttt{StateList}_n$ have already been calculated and stored, and that MAX and τ_D are constant. Model parameters for the diffusion coefficient D and voxel length ℓ are also used.

Algorithm 2 shows the details of this process. The output of this algorithm is a vector map where the positions correspond to voxel indices and the values correspond to the number of particles that have traveled to that voxel from the originating voxel n_k via diffusion after an interval of length τ_D seconds.

Algorithm 2 DFSP diffusion step with splitting

INPUT: n_k, τ_D, $\texttt{TransitionMatrix}_n$, $\texttt{StateList}_n$, $\texttt{ErrorTolerance}$
OUTPUT: Output State

initialize once: $\texttt{PDVLookupTable}$, $n_{max} \leftarrow \infty$, $\mathcal{L} \leftarrow 0$
if $n_k >= n_{max}$ **then**
  return Output State $\leftarrow$ $\texttt{DFSP\_Diffusion}(\lfloor n_k/2 \rfloor, \mathcal{L}+1)$ + $\texttt{DFSP\_Diffusion}(\lceil n_k/2 \rceil, \mathcal{L}+1)$
else
  if $\texttt{PDVLookupTable}$ contains n_k **then**
    $\texttt{PDV} \leftarrow \texttt{PDVLookupTable}[n_k]$
  else
    $\texttt{PDV} \leftarrow \exp(\ \texttt{TransitionMatrix}_{n_k} \times D/\ell^2 \times \tau_D\ ) \times [1, 0, 0, \cdots, 0]^T$
    if $\texttt{PDV}[\text{end}] > \texttt{ErrorTolerance}\ /\ 2^{\mathcal{L}}$ **then**
      $n_{max} \leftarrow n_k$
      return Output State $\leftarrow$ $\texttt{DFSP\_Diffusion}(\lfloor n_k/2 \rfloor)$ + $\texttt{DFSP\_Diffusion}(\lceil n_k/2 \rceil)$
    end if
    $\texttt{PDVLookupTable}[n_k] \leftarrow \texttt{PDV}$
  end if
  Generate a random number $X \in U(0, 1)$
  Find the smallest integer μ such that $\sum_{j=1}^{\mu} \texttt{PDV}[j] > X$
  return Output State $\leftarrow \texttt{StateList}_{n_k}[\mu]$
end if

Reactions In our computational framework for reaction-diffusion problems, we use a fractional step method which simulates the diffusive transfers by DFSP and the reaction events by SSA. We begin

at t_0 and calculate the first reaction event. We simulate reactions until the time to the next reaction would advance the simulation beyond $t_0 + \tau_D$, at which point we forego the last reaction and perform a diffusion step using DFSP. After the diffusion step, the simulation is at time $t_0 + \tau_D$. This process is repeated until the simulation is complete.

This process is detailed in Algorithm 3. Inputs to this algorithm are τ_D, the stoichiometric matrix ν and the initial state of the system. The calls to `DFSP_Diffusion` use a `StateList` and `TransitionMatrix` that correspond to the geometry and jump propensities of the problem as well as a specified `ErrorTolerance`.

Algorithm 3 RDME simulation algorithm using DFSP for diffusion and SSA for reactions

initialize system state: X, $t = 0$
Calculate the propensity functions $a_{jk}(X)$ and $a_0 \leftarrow \sum_{k=1}^{K} \sum_{j=1}^{M} a_{jk}(X)$
{where M is the number of reactions and K is the number of voxels}
Generate two random numbers $r_1, r_2 \in U(0,1)$
$t_{next\_rxn} \leftarrow t + \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right)$
$t_{next\_diff} \leftarrow t + \tau_D$
while $t < t_{final}$ **do**
  if $t_{next\_rxn} < t_{next\_diff}$ **then**
    Find μ_r, μ_x smallest integers to satisfy $\sum_{k=1}^{\mu_x} \sum_{j=1}^{\mu_r} a_{jk} > r_2 a_0$
    Update $X_{\mu_x}(t_{next\_rxn}) = X_{\mu_x}(t) + \nu_{\mu_r}$
    Generate two random numbers $r_1, r_2 \in U(0,1)$
    $t \leftarrow t_{next\_rxn}$
  else
    $X_{next} \leftarrow \emptyset$
    for $k \in (1...K)$ **do**
      for $i \in (1...N)$ **do**
        $X_{next} \leftarrow X_{next} +$ `DFSP_Diffusion`$(X_{k,i})$ {diffusion of species i in voxel k}
      end for
    end for
    $X \leftarrow X_{next}$
    $t \leftarrow t_{next\_diff}$
    $t_{next\_diff} \leftarrow t + \tau_D$
  end if
  Update propensity functions $a_{jk}(X)$ and $a_0 \leftarrow \sum_{k=1}^{K} \sum_{j=1}^{M} a_{jk}(X)$
  $t_{next\_rxn} \leftarrow t + \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right)$
end while

4 Examples and Analysis

We examine two models to explore the validity, accuracy and speed of DFSP. The first is a model of pure diffusion. The second is a biologically inspired reaction-diffusion spatial stochastic model.

4.1 Diffusion Example

The first example is composed of a single chemical species diffusing in one dimension. The domain is periodic ($\Omega = 12.4\mu m$) and we discretized it into 200 voxels of length $\ell = 0.062\mu m$. This domain is equivalent to a circle with radius 2μm, so we will plot the results on the range $[-2\pi, 2\pi)$. The initial condition is a step-function such that each voxel in the range $[-2\pi, 0)$ has 100 molecules and the remaining voxels are empty. The chemical species move with a diffusion coefficient of $0.001\mu m^2 s^{-1}$. Numerical experiments show that the relaxation time of this system is approximately 7000 seconds (data not shown). In this example, we use the adaptive step splitting with $MAX = 5$. Figure 2 shows the initial condition (dashed blue), a transient state (dotted black) and a final state (solid blue) for a single sample trajectory of this model.

[Figure 2 Here]

4.1.1 Validation

To test the validity of solving the diffusion example with ISSA or DFSP we solve for the moments analytically (see Appendix A for derivation). Figure 3 shows the error in the mean and variance as a function of time for three different sized ensembles of ISSA and DFSP trajectories. The error is calculated using the L_∞ norm (across space) of the difference between the ensemble moments and the analytically derived moments, divided by the norm of the analytical moment.

$$\text{Normalized } L_\infty \text{ error}(t) = \frac{\|\text{analytical\_moment}(x,t) - \text{ensemble\_moment}(x,t)\|_\infty}{\|\text{analytical\_moment}(x,t)\|_\infty} \tag{9}$$

As the ensemble size increases, the error in both the mean and the variance decreases at the same rate for ISSA as DFSP. Figure 4 shows the error in the mean and variance as a function of voxel size.

As voxel size decreases, the error decreases. This shows convergence of RDME solution methods to the analytical solution to the stochastic diffusion equation. Since the ISSA is an exact simulation method to the RDME while DFSP is an approximate method, this analysis shows that DFSP is just as valid as the ISSA for these parameter values.

[Figure 3 here]

[Figure 4 here]

To assess the accuracy of DFSP, we treat an ensemble of ISSA simulations as the baseline distribution because the ISSA is a true realization of the RDME and its ensemble converge to the exact solution of the RDME. The Kolmogorov distance [31] is a standard measurement of the difference between two cumulative distribution functions (CDF), it is defined as the largest deviation between two CDFs. We choose this measure because it compares all the moments of two distributions and is thus a stronger tool for analysis than methods that use individual moments. We will plot the average Kolmogorov distance across space ($Kmean$) sampled at each point in time. This is given by

$$Kmean(a,b,t) \;=\; \frac{1}{N}\sum_{n=0}^{N-1} \|CDF_a(n\ell,t) - CDF_b(n\ell,t)\|_\infty \tag{10}$$

where N is the number of voxels. The $CDF_a(x,t)$ is calculated from an ensemble of trajectories generated by algorithm a (e.g. ISSA or DFSP) sampled at spatial location x at time t. We compare the $Kmean$ of two independent ISSA ensembles (this is known as the self-distance) at each sampled point in time with the $Kmean$ of an ISSA ensemble and a DFSP ensemble. If the two $Kmean$ values are similar, then DFSP is statistically indistinguishable from ISSA for this ensemble size.

Figure 5 shows $Kmean$ values across time for the ISSA self distance and ISSA versus DFSP. We show results for ISSA versus DFSP for two sets of simulation parameters: the first uses $\tau_D = 0.1s$, `ErrorTolerance` $= 10^{-5}$ and the second uses $\tau_D = 1.9s$, `ErrorTolerance` $= 10^{-3}$. These results are for an ensemble size of 10^5 trajectories. We note that for an ensemble size $\leq 10^4$ DFSP is indistinguishable from ISSA (data not shown). These results show that for a sufficiently small values of τ_D and `ErrorTolerance` DFSP is a good approximation for ISSA. For the results with $\tau_D = 1.9$s

the adaptive step splitting fails to meet the error tolerance; therefore, as the ensemble size grows the error accrued by DFSP is no longer negligible. Thus, it is clear that for increasing values of τ_D and `ErrorTolerance` the error in the simulation grows. We will show that it is possible to utilize this feature of DFSP to trade accuracy for computational performance.

[Figure 5 Here]

4.1.2 Error Analysis

To study the error properties of DFSP, we must first find the limits of our adaptive step splitting error control method. The contribution from diffusion to the total error should be constant for all values of τ_D as long as we are able to move at least one molecule per DFSP diffusion step without violating our error tolerance. Figure 6 shows a plot of the maximum possible number of molecules moved per diffusion step of DFSP for various values of τ_D and a fixed error tolerance of 10^{-5}. To find the maximum number of molecules we can move for a given τ_D we compute DFSP matrix exponentials for increasing molecule counts. The maximum number that can be moved is one less than the number at which the estimate error first exceeds the tolerance. From this study, we determined that the maximum value of τ_D is 0.925s.

[Figure 6 Here]

To measure the error in the simulated ensembles, we integrate the deviation between the K-distance of DFSP and ISSA and the self-distance of ISSA over space and time, normalized by the size of the domain.

$$\text{Error}_{\tau_D} \;=\; \frac{\int\int |\text{DFSP}_{\tau_D}(x,t) - \text{ISSA}(x,t)|\; dx\, dt}{\int\int\; dx\, dt} \tag{11}$$

where $\text{ISSA}(x,t)$ is the K-distance over space and time between two ensembles of 10,000 runs of the ISSA, and $\text{DFSP}_{\tau_D}(x,t)$ is the K-distance over space and time between 10,000 runs of the DFSP algorithms (with diffusion step τ_D) and 10,000 runs of the ISSA algorithm. We examine this error metric for varying values of τ_D with a fixed error tolerance of 10^{-5}. Figures 7 shows the error as a function of τ_D for the diffusion example as well as the G-protein example (discussed in Section 4.2).

This shows that for this range of values of τ_D, the error in the diffusion example is constant, and a function only of the `ErrorTolerance` parameter.

[Figure 7 Here]

4.2 G-protein Cycle Example

The second example is the pheromone induced G-protein cycle in *Saccharomyces cerevisiae*. We have converted the PDE model from [32] into a stochastic model and for brevity reduced it to ligand, receptor and G-protein species. The ligand level is constant in time but it varies spatially (a cosine function) with parameters determined experimentally. The ligand binds stochastically with an initially isotropic field of receptor proteins. The bound receptor activates the G-protein, causing the G_α and $G_{\beta\gamma}$ sub-units to separate. G_α acts as an auto-phosphotase and upon dephosphorylation, rebinds with $G_{\beta\gamma}$ to complete the cycle. The spatial domain is identical to the previous model and the simulation time is set to 100 seconds, as deterministic simulation shows that steady state is achieved by that time (data not shown). $G_{\beta\gamma}$ is the component farthest downstream from the ligand input and acts as signal to the downstream Cdc42 cycle and will therefore be the output for this model. Figure 8 shows the constant ligand gradient (left) and the spatial distribution of $G_{\beta\gamma}$ over 1000 runs (right, mean and standard deviation). See Appendix B for complete description of the reactions.

[Figure 8 Here]

4.2.1 Validation

For the G-protein example, Figure 9 shows the $Kmean$ for ISSA versus DFSP (using $\tau_D = 0.1s$, `ErrorTolerance` $= 10^{-5}$) for an ensemble size of 10^5 trajectories. Note that for an ensemble size of $\leq 10^3$ trajectories, DFSP is indistinguishable from ISSA. For these simulation parameters the difference between ISSA and DFSP values of $Kmean$ is constant over time, and DFSP $Kmean$ is consistent across the time span of the simulation. This indicates that the simulation is stable, but there is an error in the results that shows up as a difference between the DFSP and ISSA curves. We will discuss the source of this error and provide an analysis in the following section. We also

show results for τ_D increased to the CFL limit [33], which is $\sim 1.9s$, and `ErrorTolerance` to 10^{-3} in an attempt to determine the limits of DFSP's ability to handle full reaction-diffusion models. For these parameters the specified `ErrorTolerance` cannot be met, though the adaptive splitting moves a single molecule per step. The difference between this curve and the ISSA curve is significantly more, and is oscillatory in time. This indicates that the simulation results are inconsistent.

[Figure 9 Here]

4.2.2 Error Analysis

Over a given timestep of length τ_D we first apply the reaction operator (SSA in this case) to the system, then the diffusion operator (using FSP) is applied to the resulting state of the system. Since these operators are decoupled an additional splitting error is incurred by the method when reaction are included. Molecules that react in the timestep are not diffused, and molecules produced by a reaction in the timestep are diffused for the full length of the timestep.

DFSP applied to the RDME is an operator split method which is a first order Strang-splitting scheme [34], and as such it is expected that the error should increase approximately linearly with τ_D. Figure 7 shows the error as a function of τ_D. We see that the error in the G-protein example is increasing approximately linearly with respect to τ_D and collapses to the error in the diffusion only system as τ_D goes to zero, confirming our expectation.

4.2.3 Performance

Figure 10 shows the speedup of DFSP over ISSA and MSA for the G-protein example. The performance increase for DFSP over ISSA and MSA is due in part to the difference in the number of times the reaction propensities must be updated as a result of diffusion events. For one realization, the expected number of diffusion events in an ISSA simulation is 1.2×10^6. Thus the reaction propensities must be updated approximately 2.4×10^6 times (source and destination voxels for each diffusion event). By numerical experimentation, the average number of reaction events for any of the methods is $\geq 170,000$. The time to the next diffusion event for MSA is given as the minimum of the time

to the next reaction step and a predetermined time step, therefore there must be at least as many diffusion events in an MSA simulation (regardless of stencil) as reaction events. For MSA, diffusion is done in all voxels, therefore updates need to be done in every voxel at each time step. Therefore, the expected number of reaction propensity updates in MSA due to diffusion steps is $\geq 3.4 \times 10^7$. For DFSP with $\tau_D = 0.1$s, 1000 diffusion steps are taken, and the reaction propensities are updated in every voxel on each DFSP step, resulting in 2×10^5 reaction updates. Therefore, it is reasonable to expect that for this problem DFSP will be $\sim 10^2$ times faster than MSA and ~ 10 times faster than ISSA for $\tau_D = 0.1$ for this problem. Figure 10 validates this claim.

[Figure 10 Here]

Next we examine the effect of different spatial discretization schemes on performance. Figure 11 shows the computation time as a function of τ_D for three levels of mesh refinement, and the computation time for ISSA at each level for comparison. The computation time for DFSP does not vary as greatly as ISSA for different mesh sizes. In solving for the diffusion step of the algorithm, DFSP iterates over the voxels in the system and thus should scale linearly with number of voxel. For comparison, in ISSA the number of diffusion jumps scales as $2/\ell^2$. Further calculations show that as we double the number of voxels the runtime for DFSP doubles, while for ISSA it increases by a factor of eight. This verifies our expectations.

For N=100, ISSA outperforms DFSP for the range of τ_D values shown. However for this discretization level τ_D can be as large as 3.6s resulting in speedups proportional to the N=200 and N=400 mesh sizes. For reaction-diffusion systems the fractional step decoupling error is proportional to τ_D. Thus for coarse meshes where a large τ_D is possible, global accuracy constraints may force a parameter selection such that ISSA performs better than DFSP.

[Figure 11 Here]

5 Conclusions

DFSP is a powerful new algorithm that yields impressive performance improvements over ISSA. DFSP provides a means to quantify and control the error, allowing a precise trade-off between accuracy and performance. Additionally, unlike many hybrid algorithms, DFSP conserves mass.

As multi-core and graphics processing unit (GPU) computing becomes even more prevalent, the importance of algorithms that are able to take advantage of these new technologies will increase. DFSP avoids many of the serial limitations imposed on spatial stochastic simulation. We are currently exploring enhancements that utilize these features. Another advantage of DFSP is that it extends simply to higher dimensional systems. This will be demonstrated in our future work.

The speedup offered by DFSP enables the simulation on a workstation of ensemble sizes that were previously feasible only on high performance clusters. It extends the scope of problems that are computable on high performance clusters. To produce our validation data for the G-protein cycle model we needed an ensemble of 100,000 runs for statistical accuracy. The DFSP algorithm generated this data set in 6.2 hours (for $\tau_D = 0.1$s and error tolerance of 10^{-5}) and 3.8 hours (for $\tau_D = 1.9$s and error tolerance of 10^{-3}) on a commodity desktop workstation with a quad-core processor (computing four trajectories simultaneously). The ISSA data sets were generated on a high performance computer cluster, so direct comparison is not possible. However, we estimate that each of the ISSA data sets would take approximately 472 processor hours, or 118 real hours (approximately 5 days) to calculate on the desktop workstation.

6 Acknowledgements

We would like to thank Andreas Hellander and Dan Gillespie for their helpful comments.

References

[1] H. McAdams and A. Arkin, Proceedings of the National Academy of Sciences **94**, 814 (1997).

[2] M. Elowitz, A. Levine, E. Siggia, and P. Swain, Science **297**, 1183 (2002).

[3] D. Gillespie, Annual Review of Physical Chemistry **58**, 35 (2007).

[4] M. Howard and A. D. Rutenberg, Physical Review Letters **90**, 128102 (2003).

[5] D. Fange and J. Elf, PLoS Computational Biology **2**, e80 (2006).

[6] K. Doubrovinski and M. Howard, Proceedings of the National Academy of Sciences **102**, 9808 (2005).

[7] J. Elf and M. Ehrenberg, Systems Biology, IEE Proceedings **1**, 230 (2004).

[8] B. Munsky and M. Khammash, The Journal of Chemical Physics **124**, 044104 (2006).

[9] C. W. Gardiner, K. J. McNeil, D. F. Walls, and I. S. Matheson, Journal of Statistical Physics **14**, 307 (1976).

[10] N. Shnerb, Y. Louzoun, E. Bettelheim, and S. Solomon, Proceedings of the National Academy of Sciences **97**, 10322 (2000).

[11] S. Isaacson, SIAM Journal on Applied Mathematics **70**, 77 (2009).

[12] R. Erban and S. Chapman, ArXiv e-prints (2009).

[13] M. Gibson and J. Bruck, The Journal of Physical Chemistry A **104**, 1876 (2000).

[14] J. Hattne, D. Fange, and J. Elf, Bioinformatics **21**, 2923 (2005).

[15] T. T. Marquez-Lago and K. Burrage, The Journal of Chemical Physics **127**, 104101 (2007).

[16] S. Lampoudi, D. Gillespie, and L. Petzold, The Journal of Chemical Physics **130**, 094104 (2009).

[17] Z. Zheng, R. Stephens, R. Braatz, R. Alkire, and L. Petzold, The Journal of Computational Physics **227**, 5184 (2008).

[18] J. Rodriguez, J. Kaandorp, M. Dobrzynski, and J. Blom, Bioinformatics **22**, 1895 (2006).

[19] L. Ferm, A. Hellander, and P. Lötstedt, Technial Report 2009-010, Dept of Information Technology, Uppsala University (2009).

[20] S. Engblom, L. Ferm, A. Hellander, and P. Lötstedt, SIAM Journal of Scientific Computing **31**, 1774 (2009).

[21] D. Gillespie, Physica A: Statistical Mechanics and its Applications **188**, 404 (1992).

[22] D. Gillespie, The Journal of Physical Chemistry **81**, 2340 (1977).

[23] Y. Cao, H. Li, and L. Petzold, The Journal of Chemical Physics **121**, 4059 (2004).

[24] A. Slepoy, A. Thompson, and S. Plimpton, The Journal of Chemical Physics **128**, 205101 (2008).

[25] H. Li and L. Petzold, International Journal of High Performance Computing (2009).

[26] D. Gillespie, The Journal of Chemical Physics **115**, 1716 (2001).

[27] S. Peleš, B. Munsky, and M. Khammash, The Journal of Chemical Physics **125**, 204104 (2006).

[28] S. McNamara, K. Burrage, and R. Sidje, International Journal of Computational Science **2**, 402 (2008).

[29] B. Munsky and M. Khammash, Journal of Computational Physics **226**, 818 (2007).

[30] K. Burrage, M. Hegland, S. MacNamara, and R. Sidje, A krylov-based finite state projection algorithm for solving the chemical master equation arising in the discrete modeling of biological systems, in *Proceedings of the 150th Markov Anniversary Meeting*, edited by A. Langville and W. Stewart, page 2138, Boson Books, 2006.

[31] A. Kolmogorov, IEEE Transactions on Information Theory **IT-14**, 662 (1968).

[32] C.-S. Chou, Q. Nie, and T.-M. Yi, PLoS ONE **3**, e3103 (2008).

[33] R. Courant, K. Friedrichs, and H. Lewy, Mathematische Annalen **100**, 32 (1928).

[34] G. Strang, SIAM Journal on Numerical Analysis **5**, 506 (1968).

Appendices

A Analytical solution to the diffusion example

On a one-dimensional infinite domain, if a single molecule is homogeneously distributed in the interval $[a, b)$ then its probability distribution function $p(x, t)$ is a step function

$$p_0(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{else.} \end{cases} \tag{12}$$

We can evolve the probability distribution forward in time by solving the diffusion equation

$$\frac{\partial p(x,t)}{\partial t} = D\,\nabla^2 p(x,t) \tag{13}$$

using (12) as the initial condition. The solution to the diffusion equation on a one-dimensional infinite domain is the convolution of the initial condition with a Gaussian kernel:

$$p(x,t) = \frac{1}{\sqrt{2\pi D t}} \int_a^b \frac{1}{b-a} \exp\left(\frac{-(x_0 - x)^2}{2Dt}\right) dx_0. \tag{14}$$

Since we seek to compare against numerical solutions solved on a discretized domain, we can use (14) to find the probability that a single molecule homogeneously distributed in the interval $[a, b)$ (starting voxel) at $t = 0$ will be in the interval $[x_1, x_2)$ (ending voxel) at time t:

$$P(x_1,\, x_2, t\,|a, b) = \frac{1}{\sqrt{2\pi D t}} \int_{x_1}^{x_2} \int_a^b \frac{1}{b-a} \exp\left(\frac{-(x_0 - x)^2}{2Dt}\right) dx_0\, dx. \tag{15}$$

For the solution on the domain $[-2\pi,\, 2\pi)$ with periodic boundary conditions, we can use mirroring. Mirroring is a method of translating a periodic domain into an infinite domain by repeating the initial condition function periodically from $(-\infty, \infty)$. The problem is then solved by integrating

across the infinite domain for solutions at points in the original domain:

$$P(x_1,\, x_2, t\,|a, b) = \sum_{j=-\infty}^{\infty} \frac{1}{\sqrt{2\pi Dt}} \int_{x_1}^{x_2} \int_{a+4\pi j}^{b+4\pi j} \frac{1}{b-a} \exp\left(\frac{-(x_0 - x)^2}{2Dt}\right) dx_0\, dx. \tag{16}$$

For finite precision of our answer we need take only a finite number of terms

$$P(x_1,\, x_2, t\,|a, b) = \sum_{j=-J}^{J} \frac{1}{\sqrt{2\pi Dt}} \int_{x_1}^{x_2} \int_{a+4\pi j}^{b+4\pi j} \frac{1}{b-a} \exp\left(\frac{-(x_0 - x)^2}{2Dt}\right) dx_0\, dx \tag{17}$$

where $2J + 1$ is a sufficient number of terms for the required precision of our solution.

For the diffusion example, Figure 2, the step function initial conditions is equivalent to homogeneously distributing 100 molecules in each of the 100 voxels in the interval $[-2\pi, 0]$ (for $\ell = 0.06\mu m$). If a molecule starts in the kth voxel, and we want to know the probability that it will be within a given voxel containing the interval $[x, x + \ell)$ at time t we can use (17) with the following inputs:

$$P_k(x, t) \;=\; P\left(x,\, x + \ell,\, t\,|\, -2\pi + (k-1)\ell, -2\pi + k\ell\right). \tag{18}$$

To compare to our numerical solutions we need to find the moments of the population of molecules in a given voxel at a given time. The generic binomial distribution is a sum of many independent Bernoulli trials and the mean and the variance of such a distribution are equal to the sums of the means and variances of each individual trial. A molecule being located within a given voxel at time t is a Bernoulli trial with probability given by (17) because it is either within the voxel or it is not. Thus the population $u(x, t)$ of a voxel containing the interval $[x, x + \ell)$ at any time t is binomially distributed, and the analytical solution for the mean and variance of the population is given by

$$E[u(x, t)] = \sum_{k=1}^{100} n_k\, P_k(x, t) \tag{19}$$

$$Var[u(x, t)] = \sum_{k=1}^{100} n_k\, P_k(x, t)\;\; [1 - P_k(x, t)] \tag{20}$$

where n_k is the initial population of the voxel containing the interval $[x, x + \ell)$.

We can use equations (19) and (20) to compare the mean and variance obtained from an ensemble of runs from ISSA and DFSP.

B G-protein cycle example

These are the equations that describe the reactions of the G-protein cycle example.

Equations

$$\emptyset \quad \overrightarrow{k_{Rs}} \quad [\mathrm{R}] \tag{21}$$

$$[\mathrm{R}] \quad \overrightarrow{k_{Rd0}} \quad \emptyset \tag{22}$$

$$[\mathrm{L}] + [\mathrm{R}] \quad \overrightarrow{k_{RL}} \quad [\mathrm{RL}] + [\mathrm{L}] \tag{23}$$

$$[\mathrm{RL}] \quad \overrightarrow{k_{RLm}} \quad [\mathrm{R}] \tag{24}$$

$$[\mathrm{RL}] \quad \overrightarrow{k_{Rd1}} \quad \emptyset \tag{25}$$

$$[\mathrm{RL}] + [\mathrm{G}] \quad \overrightarrow{k_{Ga}} \quad [\mathrm{Ga}] + [\mathrm{Gbg}] + [\mathrm{RL}] \tag{26}$$

$$[\mathrm{Ga}] \quad \overrightarrow{k_{Gd}} \quad [\mathrm{Gd}] \tag{27}$$

$$[\mathrm{Gd}] + [\mathrm{Gbg}] \quad \overrightarrow{k_{G1}} \quad [\mathrm{G}] \tag{28}$$

Rate Constants

$k_{RL} = 2e-03\ \mathrm{M}^{-1}, \quad k_{RLm} = 1e-02, \quad k_{Rs} = 4/SA, \quad k_{Rd0} = 4e-04, \quad k_{Rd1} = 4e-04,$

$k_{G1} = 1 \times SA, \quad k_{Ga} = 1e-05 \times SA, \quad k_{Gd} = 0.1,$

$D_m = 0.001\,\mu m^2/s$

Total Populations and Initial Conditions:

$$SA = 50.2655\,\mu m^2, \quad V = 33.5\,\mu m^3,$$

$$[R]_0 = 10000/SA, \quad [G]_0 = 10000/SA,$$

$$[L](z) = L_{mid} + L_{slope}(z - z_0),$$

$$L_{mid} = 2\,nM, \quad L_{slope} = 1\,nM(\mu m)^{-1}$$

Figures

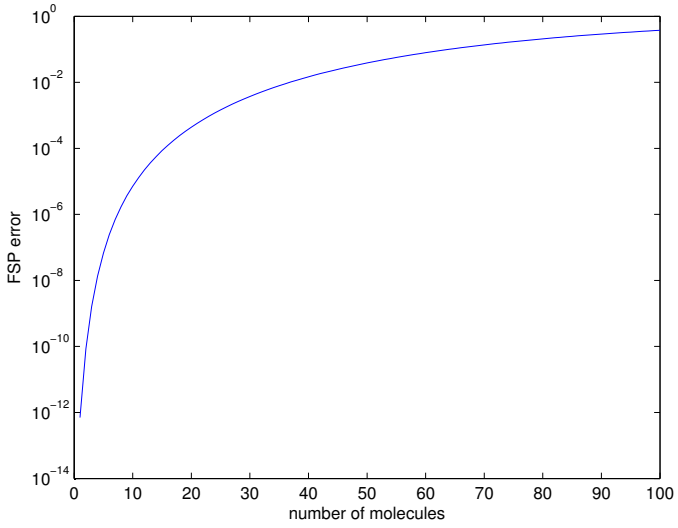

Figure 1: Projection error (ϵ) for varying number of molecules given $\tau_D = 0.1s$, $\ell = 0.62\mu m$, $D = 0.001\mu m^2 s^{-1}$. Adaptive step splitting allows us to take advantage of the independence of diffusing molecules and the nonlinear relationship of projection error to the number of molecules diffused to reduce the total error of diffusion step over τ_D by splitting it into sub-sub-problems of fewer molecules rather than by splitting the time step.

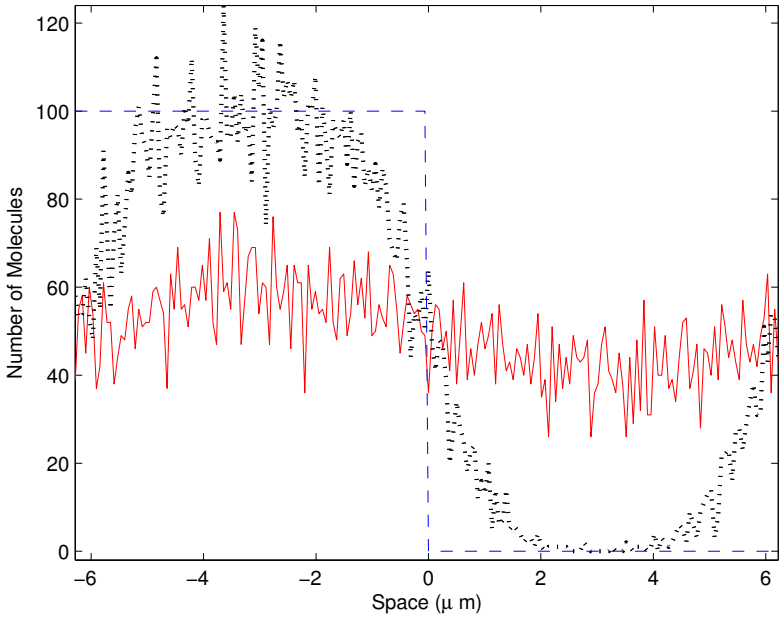

Figure 2: Solution to a pure diffusion problem with a step function as the initial condition. Plotted is the state of the system at t = 0s (dashed blue), 500s (dotted black) and 7000s (solid red) for a stochastic trajectory. The domain is a circle with radius $2\mu m$, subdivided equally into 200 voxels.

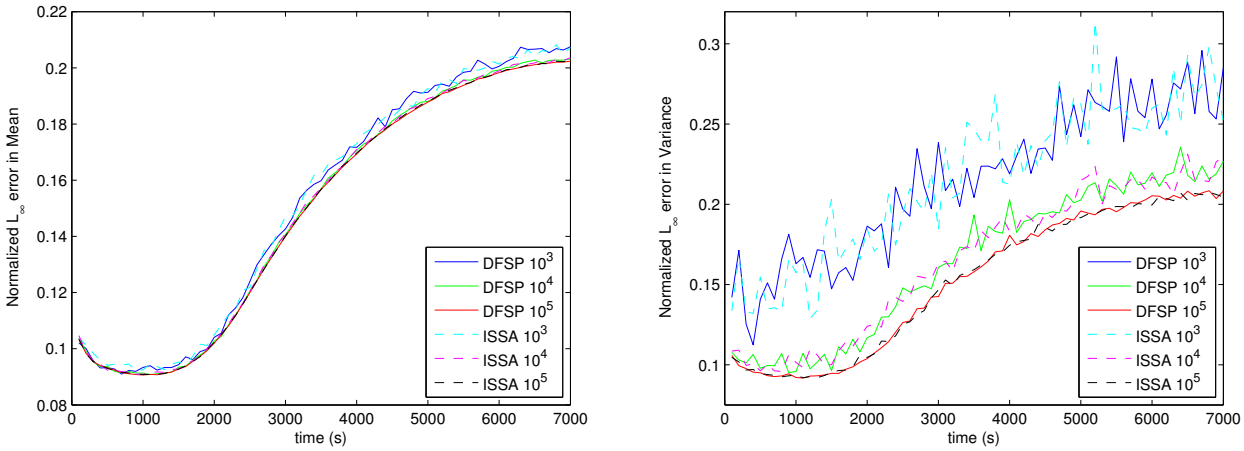

Figure 3: Plot of the Normalized L_∞ error (maximum deviation from analytical solution) versus time in the mean (left) and variance (right) for varying ensemble sizes for both DFSP ($\tau_D = 0.1$s) and ISSA (voxel size of $0.06\mu m$) for an ensemble size of 10^3 trajectories. The error increases with time (as expected for a discretized solution) at the same rate for both DFSP and ISSA. Additionally, the error decreases (to the discretization error limit) with increasing ensemble size at the same rate for DFSP and ISSA.

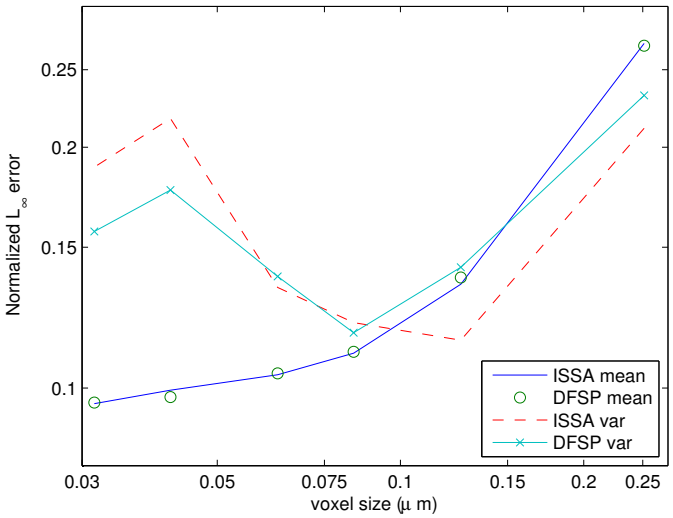

Figure 4: Plot of the Normalized L_∞ error versus voxel size (both on a log-scale) calculated at $t = 100$s (a transient state) for an ensemble size of 10^3 trajectories. As voxel size decreases, the error in the mean decreases at the same rate for both DFSP and ISSA. The error in the variance shows a similar trend, however it also shows increased error for small voxel sizes. This is mostly likely sampling error due to a constant system population distributed into an increasing number of voxels.

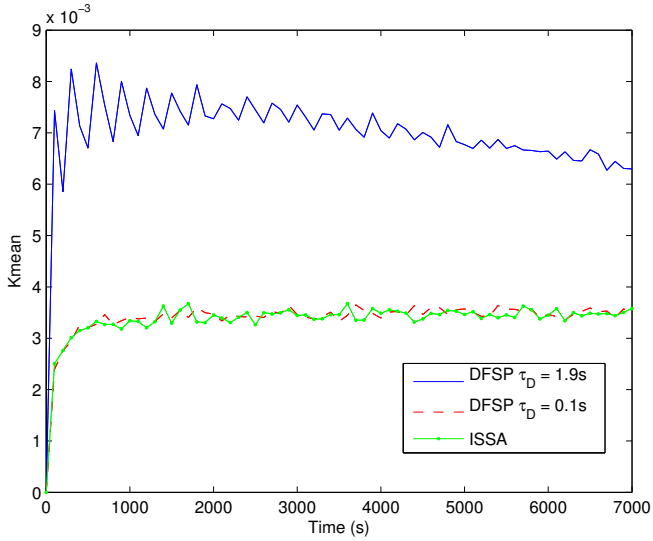

Figure 5: The distribution distance for the diffusion example, the $Kmean$ at every 100s for 7000s for ISSA-vs-ISSA (green line with dots), ISSA-vs-DFSP with $\tau_D = 0.1$s (dashed red line) and $\tau_D = 1.9$s (blue line). In this plot the ensemble size is 10^5, at which point the DFSP solution becomes distinguishable from the ISSA solution for larger time steps. For ensemble sizes $\leq 10^4$, the DFSP solution is indistinguishable from the ISSA solution for both step sizes (results not shown).

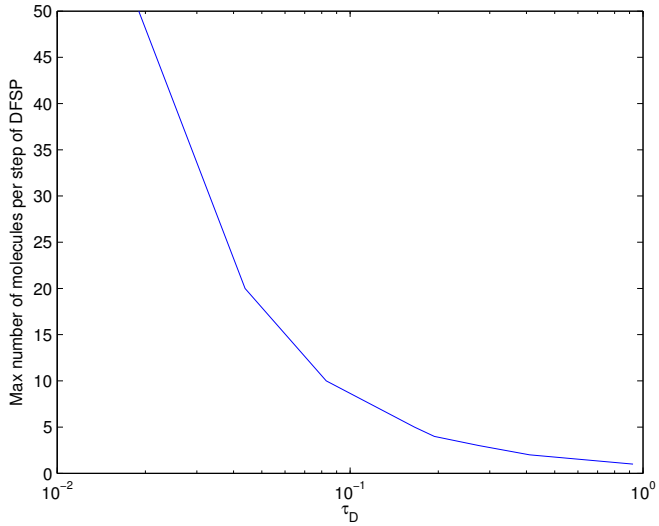

Figure 6: Maximum number of molecules moved in a single diffusion step of DFSP versus τ_D for an error tolerance of 10^{-5}. Values of $\tau_D > 0.925$s will try to move less than one molecule and thereby violate the error tolerance.

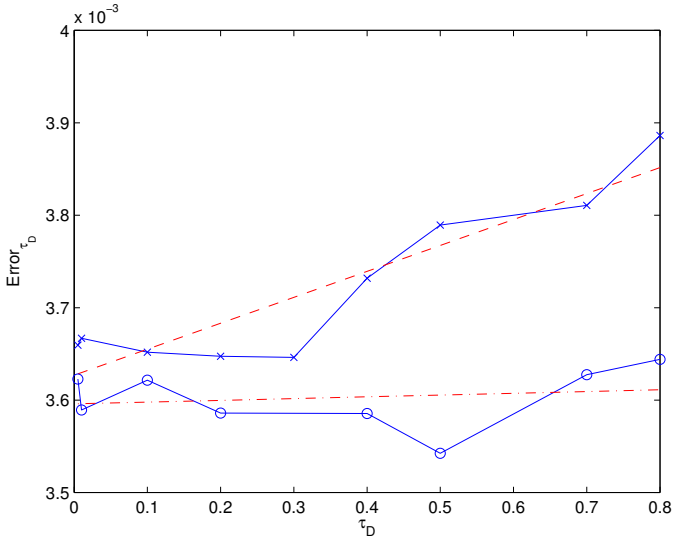

Figure 7: Error$_{\tau_D}$ vs τ_D for the diffusion example (line with circles) and G-protein cycle example (line with x) and linear fits (dot-dashed line and dashed line respectively). For the diffusion example the error is constant and only a function of the error tolerance, this is because there is no contribution to the error from the reaction operator. For the G-protein cycle example the error increases linearly with τ_D and converges to the diffusion error as τ_D goes to zero. This is because the error in the reaction operator is linear with the timestep (τ_D); as the timestep goes to zero the reaction error goes to zero.

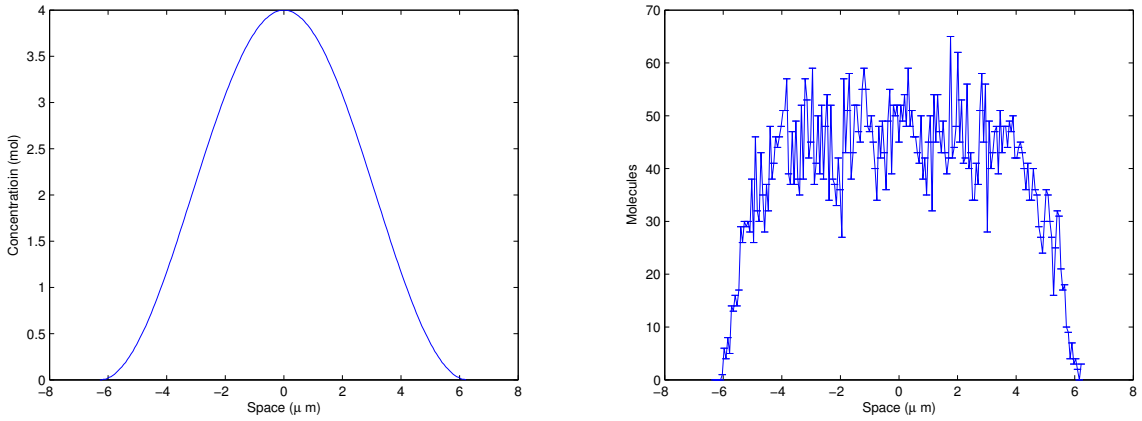

Figure 8: Spatial concentration of Ligand (left) and mean and variance of $G_{\beta\gamma}$ molecules count (right) at $t = 100$s for a trajectory of the G-protein cycle example. The Ligand gradient is the input to this model and is constant in time. $G_{\beta\gamma}$ is the output and is time varying.

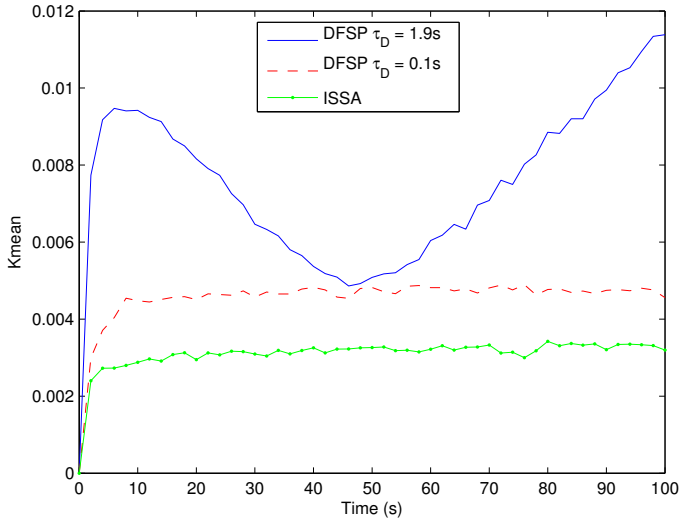

Figure 9: The distribution distance for the G-Protein cycle example, the $Kmean$ from (10) at every 100s for 7000s for ISSA-vs-ISSA (green line with dots), ISSA-vs-DFSP with $\tau_D = 0.1$s (dashed red line) and $\tau_D = 1.9$s (blue line). In this plot the ensemble size is 10^5, at which point the DFSP solution becomes distinguishable from the ISSA solution for larger time steps. For ensemble sizes $\leq 10^3$, the DFSP solution is indistinguishable from the ISSA solution for both step sizes (results not shown).

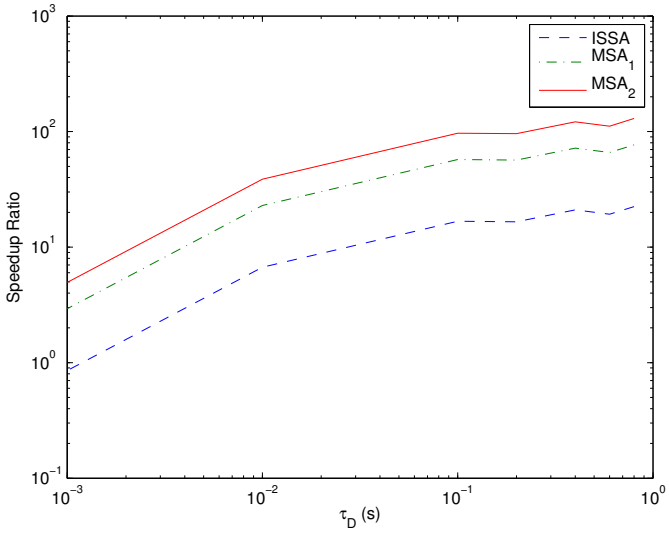

Figure 10: Speedup of DFSP over ISSA and both MSA stencils (subscript 1 denotes the case where net diffusion is solved for adjacent voxels, and 2 the case where net diffusion is solved for the two nearest neighbors on either side) for varying values of τ_D for the G-protein cycle example. DFSP presents significant performance increases over ISSA and both MSA stencils for reaction-diffusion simulation. The speedup is due in part to the number of times the reaction propensity function needs to be updated due to diffusive transfers. DFSP updates less often than ISSA or MSA. As τ_D increases, the number of updates decreases and performance increases.

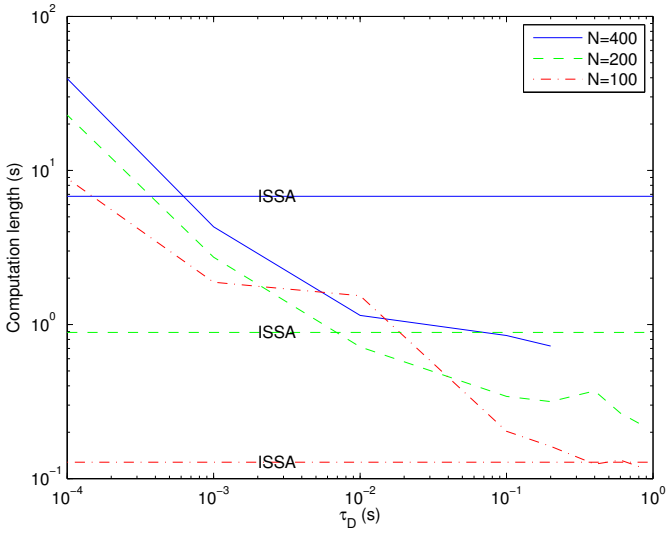

Figure 11: Computation time for DFSP with varying values of τ_D and varying numbers of voxels in the system with error tolerance of 10^{-5} for the diffusion example. ISSA computation times for each of the system sizes is provided for comparison.