Algorithms and Software for Stochastic Simulation of Biochemical Reacting Systems *

Hong Li†, Yang Cao‡, Linda R. Petzold§, Daniel T. Gillespie ¶

April 18, 2007

Abstract

Traditional deterministic approaches for simulation of chemically reacting systems fail to capture the randomness inherent in such systems at scales common in intracellular biochemical processes. In this article we briefly review the state of the art in discrete stochastic and multiscale algorithms for simulation of biochemical systems and we present the StochKit software toolkit.

1 Introduction

The time evolution of well stirred chemically reacting systems has traditionally been simulated by solving a set of coupled ordinary differential equations (ODEs). Although the deterministic formulation is sufficient in many cases, it fails to capture the inherent stochasticity in many biochemical systems formed by living cells,[1–4] in which the small population of a few critical reactant species can result in discrete and stochastic behavior. The dynamics of those systems can be simulated accurately using the machinery of Markov process theory, specifically the stochastic simulation algorithm (SSA) of Gillespie.[1,2] SSA, an essentially exact procedure for generating realizations of the chemical master equation (CME), is in widespread use for the stochastic simulation of biochemical systems. But for many real biochemical systems, the computational cost of simulation by SSA can be prohibitively high. This is due to the fact that SSA must simulate every reaction event. When there are large populations of some chemical species, and/or fast reactions involved in the system, a great many reaction events must be simulated.

*This work was supported by the California Institute of Technology under DARPA Award No. F30602-01-2-0558, by the U. S. Department of Energy under DOE award No. DE-FG03-00ER25430, by the NIH under awards GM078993 and GM075297, and by the Institute for Collaborative Biotechnologies through grant DAAD19-03-D-0004 from the U. S. Army Research Office.

†Department of Computer Science, University of California Santa Barbara. CA 93106. email: hongli@cs.ucsb.edu.

‡Department of Computer Science, Virginia Tech. VA 24061. email: ycao@cs.vt.edu

§Department of Computer Science, University of California Santa Barbara. CA 93106. email: petzold@cs.ucsb.edu.

¶Dan T. Gillespie Consulting, 30504 Cordoba Place. Castaic, CA 91384. email: GillespieDT@mailaps.org

It is very important to have a fast SSA implementation. This algorithm is likely to be a part of any accelerated and multiscale discrete stochastic methodology. A number of authors have proposed successively more efficient formulations of SSA. The Next Reaction Method (NRM),[5] Optimized Direct Method (ODM),[6] Sorting Direct Method (SDM),[7] fast Kinetic Monte Carlo Method (KMC),[8] KMC with minimal searching[9] and Logarithmic Direct Method (LDM)[10] with sparse matrix state update are mathematically equivalent formulations which offer improved performance over the original SSA. Often SSA is used to generate ensembles of stochastic realizations from which approximate probability density functions for species populations can be obtained. Parallel computation has been used to speed up the calculation of such ensembles.[11] Some of our current work and others[12] explores the use of novel computer architectures to speed up SSA simulation. However, there is a limit to how much SSA can be sped up, given that it must simulate every reaction event in the system and there are usually a great many reaction events.

Approximate accelerated stochastic methods[13–17] have been developed to speed up the stochastic simulation. The first such method is *tau-leaping*.[13] Tau-leaping advances the simulation by a pre-selected time τ which can often be chosen large enough to encompass more than one reaction event. Stiffness (the presence of both fast and slow reactions) is often an issue in approximate discrete stochastic simulation of chemically reacting systems, just as it is an important consideration in the deterministic simulation of chemically reacting systems. To improve the numerical stability for stiff problems, *implicit tau-leaping*[18] and the *trapezoidal tau-leaping methods*[19] have been proposed. Another, complementary means of accelerating discrete stochastic simulation is the use of hybrid methods,[14,20] which approximate the fast reactions involving species with large populations via ODEs, and the slow reactions involving species with small populations via SSA. Although hybrid methods can be quite effective in solving some problems, they cannot efficiently handle the situation of a system with fast reactions involving a chemical species for which the population is small. The slow-scale SSA (ssSSA)[15] was developed to deal with this type of situation. The ssSSA makes use of a stochastic partial equilibrium approximation to advance the system at the scale of the slow reactions.

There is a great need for reliable and efficient software that makes these state of the art simulation tools available to the systems biology community. To this end, we have developed StochKit, a software toolkit for discrete stochastic and multiscale simulation of chemically reacting systems. The aim of StochKit is to make reliable and efficient stochastic and multiscale simulation accessible to practicing biologists and chemists, while remaining open to extension by accommodating new algorithms and implementations.

In this paper we review the recent work on algorithms for discrete stochastic and multiscale simulation of biochemical reaction networks. We introduce StochKit and briefly describe some early success stories with StochKit.

2 Discrete Stochastic Simulation Methods

Our concern here is with a system of molecules of N chemical species $\{S_1, \ldots, S_N\}$ which interact through M chemical reactions channels $\{R_1, \ldots, R_M\}$. We assume the system to be "well-stirred," and confined to some constant volume Ω at a constant temperature. The

state of the system at time t is specified by the vector $\mathbf{X}(t) \equiv (X_1(t), \ldots, X_N(t))$, where $X_i(t)$ is the number of S_i molecules in the system at time t.

Each reaction channel R_j is assumed to be characterized mathematically by two quantities. The first is its *state-change vector* $\boldsymbol{\nu}_j \equiv (\nu_{1j}, \ldots, \nu_{Nj})$, where ν_{ij} is the change in the S_i molecular population caused by one R_j reaction; i.e., R_j causes the system to jump, essentially instantaneously, from its present state $\mathbf{x}$ to state $\mathbf{x}+\boldsymbol{\nu}_j$. The other characterizing quantity for R_j is its *propensity function* a_j. This is defined so that $a_j(\mathbf{x})\,dt$ is the probability, given $\mathbf{X}(t) = \mathbf{x}$, that one R_j reaction will occur somewhere inside the system in the next infinitesimal time interval $[t, t+dt)$. For a reaction of the form $S_1 \to$ product(s), $a_j(\mathbf{x})$ typically has the form $c_j x_1$, where c_j is some constant. For a reaction of the form $S_1 + S_2 \to$ product(s), $a_j(\mathbf{x})$ typically has the form $c_j x_1 x_2$, or $c_j \frac{1}{2} x_1(x_1 - 1)$ if $S_2 = S_1$, where again c_j is some constant.

2.1 The SSA and its Various Formulations

The basic procedure for generating trajectories or "realizations" of $\mathbf{X}(t)$ is called the *stochastic simulation algorithm* (SSA).[1,2] The theoretical basis for the SSA is the function $p(\tau, j \,|\, \mathbf{x}, t)$, which is defined so that $p(\tau, j \,|\, \mathbf{x}, t)\,d\tau$ is the probability, given $\mathbf{X}(t) = \mathbf{x}$, that the *next* reaction in the system will occur in the infinitesimal time interval $[t+\tau, t+\tau+d\tau)$, *and* will be an R_j reaction. This function is thus the joint probability density function of the two random variables "time to the next reaction" (τ) and "index of the next reaction" (j), given that the system is currently in state $\mathbf{x}$. It is not hard to show, by applying the laws of probability, that[1]

$$p(\tau, j \,|\, \mathbf{x}, t) = a_j(\mathbf{x})\; e^{-a_0(\mathbf{x})\,\tau} = a_0(\mathbf{x})\; e^{-a_0(\mathbf{x})\,\tau} \times \frac{a_j(\mathbf{x})}{a_0(\mathbf{x})}, \tag{1}$$

where $a_0(\mathbf{x}) \equiv \sum_{k=1}^{M} a_k(\mathbf{x})$. The last form here implies that τ is an exponential random variable with mean $1/a_0(\mathbf{x})$, and j is a statistically independent integer random variable with point probabilities $a_j(\mathbf{x})/a_0(\mathbf{x})$.

There are several statistically equivalent formualtions of SSA for generating samples of τ and j according to these distributions. Perhaps the simplest is the so-called *direct method* (DM):[1,2] We draw two random numbers r_1 and r_2 from the uniform distribution in the unit-interval and take

$$\tau = \frac{1}{a_0(\mathbf{x})} \ln\left(\frac{1}{r_1}\right), \tag{2a}$$

$$j = \text{the smallest integer satisfying } \sum_{k=1}^{j} a_k(\mathbf{x}) > r_2\, a_0(\mathbf{x}). \tag{2b}$$

Therefore, for the system in state $\mathbf{x}$ at time t, we can advance the system to the next reaction by computing τ and j according to these formulas and then replacing $t \leftarrow t+\tau$ and $\mathbf{x} \leftarrow \mathbf{x} + \boldsymbol{\nu}_j$.

There are several other exact algorithms for moving the system forward in time to the next reaction according to the joint density function (1). The *first reaction method* (FRM)[1,2] generates a putative time $\tau_k = a_k^{-1}(\mathbf{x}) \ln(1/r_k)$ to the next R_k event for $k = 1, \ldots, M$, and

then chooses τ to be the smallest of the τ_k's, and j the index of the smallest. This method is usually slower than the DM because it requires Munit-interval uniform random numbers r_k. Execution time is in fact the main limitation of the SSA, because generating every reaction event one at a time can be very time-consuming if, as is usually the case, an enormous number of reaction events occurs during the times of interest. But since most accelerated "multiscale" or "hybrid" methods use the SSA as a core component, it is important to make the SSA as efficient as possible.

The *next reaction method* (NRM)[5] is a heavily modified version of the FRM that in some cases is more efficient than the DM. It essentially saves the putative next firing times of all reaction channels in an indexed binary tree priority queue, and it uses a dependency graph to determine which propensities need to be updated after each reaction. However, the maintenance overhead for these data structures limits the benefits of the FRM to very large, very loosely coupled systems. The *optimized direct method* (ODM)[6] increases the efficiency of the reaction-selection step (2b), a key bottleneck in the DM, by pre-ordering the reactions so that those with larger propensity functions have smaller index values. The *sorting direct method* (SDM)[7] carries this strategy one step further by dynamically re-ordering the reactions via a bubble-up sorting method, in which each time a reaction fires it swaps indices with the next lower-indexed reaction. Formally, the computation times of both the ODM and the SDM are $O(M)$.

SSA may be viewed as a special kind of Kinetic Monte Carlo algorithm which is applied to well-mixed chemically reacting systems. Since SSA is always applied to well-mixed chemically reacting systems, it has been possible to use the structure of these problems to put it on a solid theoretical foundation. A fast algorithm for KMC has been proposed[9] which fixes the search depth to be $\mathrm{O}(\log M)$ by reusing the intermediate data during the calculation of a_0. This algorithm is independent of the order of reactions. Further efficiency can be achieved in the implementation by using sparse matrix techniques in the system state update stage.[10]

2.2 Tau-leaping and Beyond

Significant speedups in the SSA will inevitably involve *approximations* of one kind or another. One prominent approximate acceleration procedure is the *tau-leaping* simulation method,[13] which also provides the theoretical connection between the SSA and the deterministic ODEs of traditional chemical kinetics.[21] The basic idea of tau-leaping is to advance the system by a *pre-selected* time τ during which many reactions occur. To do this accurately, we must choose τ small enough that no propensity function changes "appreciably" during τ. To the extent that this *leap condition* is satisfied, then given the system in state $\mathbf{x}$ at time t, the number of times that reaction R_j will fire in $[t, t+\tau)$ will be approximately $\mathcal{P}_j\left(a_j(\mathbf{x})\tau\right)$, the Poisson random variable with mean (and variance) $a_j(\mathbf{x})\tau$. This leads to the basic (approximate) update formula for tau-leaping:

$$\mathbf{X}(t+\tau) \doteq \mathbf{x} + \sum_{j=1}^{M} \mathcal{P}_j\left(a_j(\mathbf{x})\tau\right)\boldsymbol{\nu}_j. \tag{3}$$

Practical considerations cause the actual implementation of tau-leaping to be more complicated than simply substituting into formula (3) the current state $\mathbf{x}$ and generating M Poisson

random numbers with the indicated means. First is the problem of choosing the leap time τ. The latest recipe for doing this[22] efficiently estimates the largest τ for which the expected fractional change in every propensity function will be bounded by a user-specified amount ε (typically $\varepsilon = 0.04$). This is tricky to do since the change induced in any propensity function during a leap by τ will be a random variable, so the estimation procedure needs to take account of not only the mean of that change but also the fluctuations in that change about its mean. Second is the problem of ensuring that no reactant population be driven negative in a leap. This problem was originally attributed to the unbounded character of the Poisson random variable; however it was subsequently found to be more often a consequence of two or more reaction channels independently consuming the same reactant. One resolution of this problem[23] is to classify as "critical" all reactions that are within a user-specified number n_c firings of exhausting some reactant, and then to allow *no more than one* firing of a critical reaction during a leap (typically $n_c = 10$). This procedure[23] has the additional advantage that, as all reactions become critical, or equivalently as $n_c \to \infty$, it segues smoothly to the exact SSA. This is useful since (3) as it stands does not segue to the SSA in a computationally efficient way.

The foregoing "explicit" tau-leaping procedure has been shown to work well provided the time-scales of all the reactions are not too different. But it will seem very slow when applied to systems that evolve on widely separated timescales, because the leap condition generally restricts τ to the smallest (fastest) timescale. Much work has been devoted recently to finding approximate workarounds for such systems. For overtly "stiff" systems – systems for which the fastest evolving modes are stable – a variety of methods has been proposed. The *implicit tau-leaping* method[18] is a generalization of the implicit Euler method for handling stiff ordinary differential equations. It basically replaces (3) with

$$\mathbf{X}(t+\tau) \doteq \mathbf{x} + \sum_{j=1}^{M} \left[\mathcal{P}_j\left(a_j(\mathbf{x})\tau\right) - a_j(\mathbf{x})\tau + a_j(\mathbf{X}(t+\tau))\tau\right] \boldsymbol{\nu}_j. \tag{4}$$

This formula must be solved implicitly for the state at time $t+\tau$; usually this is done numerically using Newton's method. A slight modification of (4) will keep the state of the system on the integer lattice. Implicit tau-leaping avoids the stability limitations of explicit tau-leaping, and thus allows much larger values of τ to be used. Its main drawback is that it overdamps the natural fluctuations of the *fast* species, and requires that a strategy called "down-shifting" be applied to recover those fast species fluctuations whenever required. The *trapezoidal tau-leaping method*[19] puts a factor of $1/2$ in front of the last two terms in brackets in (4). It has been shown to improve simulation accuracy for some systems. The most recent improvement in tau-leaping is the *adaptive explicit-implicit tau-leaping* procedure,[24] which automatically identifies when stiffness is present and dynamically chooses between the explicit and implicit tau-leaping schemes.

Another approach to handling stiff systems involves a stochastic generalization of the quasi-steady state and partial equilibrium methods of deterministic chemical kinetics.[14–16,20,25–31] The theoretical basis for these methods is captured most comprehensively by the *slow-scale SSA*[15,27,31] and its practical implementation as the *multiscale SSA*.[16] The slow-scale SSA is a systematic procedure for partitioning the system into fast and slow reactions and species, and then simulating *only the slow reactions*, using specially modified propensity functions.

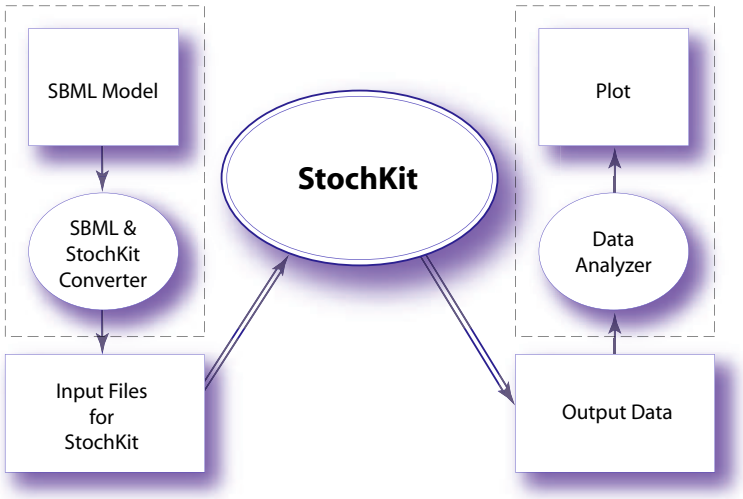

Figure 1: Simulation Process of StochKit

Implementing this procedure requires one to estimate, either theoretically or numerically, certain very specific features of the process consisting of the fast species undergoing *only* the fast reactions – the so-called the "virtual fast process". When this can be done with sufficient accuracy, enormous speedups can be achieved for very stiff systems without appreciable loss of accuracy. The important Michaelis-Menten enzyme-substrate reaction typically falls within the provenance of the slow-scale SSA.

3 StochKit– Stochastic Simulation Toolkit

3.1 What is StochKit

StochKit is an efficient, extensible stochastic simulation toolkit developed in `C++` that aims to make state of the art stochastic simulation algorithms accessible to biologists and chemists, while remaining open to extension via new stochastic and multiscale algorithms. StochKit consists of a suite of software for stochastic simulation. The StochKit core implements the simulation algorithms. Additional tools are provided for the convenience of the simulation and analysis. A typical simulation process of StochKit is shown in Figure 3.1

The intended audience for StochKit can be divided into two distinct groups: those doing research on and development of stochastic simulation methods, and those seeking to employ such methods to further their biological or chemical research. Thus the package is designed to be both simple to use and easy to extend. StochKit is freely available for download at www.engr.ucsb.edu/∼cse.

3.2 The Core Package of StochKit

The StochKit core implements state of the art stochastic simulation algorithms through a unified interface. These algorithms include Gillespie's SSA algorithm,[1,2] the optimized SSA algorithm,[6] tau-leaping methods (explicit tau-leaping,[13] implicit tau-leaping[18] and trapezoidal tau-leaping[19] with fixed stepsize, and adaptive stepsize, nonnegativity-preserving explicit tau-leaping method[22,23]), slow scale SSA[15] and multiscale SSA[16] methods. These algorithms were implemented as modules. Users who are not experts in stochastic simulation can simply use the default option (1 for SSA and 0 for adaptive explicit tau-leaping method) to let StochKit automatically select the simulation algorithm and the corresponding stepsize. Advanced users can select different options for better performance or accuracy. Those developing new algorithms (or simply refining existing ones) need only supply a new module that captures their particular innovation (i.e. stepsize selection, single step execution, data management, etc.). Basic matrix and vector operations are provided in the Math library of StochKit, which facilitates the development of new modules. Examples are provided to explain how to use and extend these algorithms (see[11]). The capacity to run an ensemble of stochastic simulations is provided.

Besides the convenience of usage and extension, special attention has been paid to the accuracy of simulation results by StochKit. An essential concern is the pseudorandom number generator. High quality pseudorandom number generation is the cornerstone of any stochastic simulation system. In fact, statistical results can only be relied upon if the independence of the samples can be guaranteed. Thus, a high quality pseudorandom number generator is crucial for StochKit. The standard library routines `rand()` from `C` could in principle be used for a uniform random number generator in our simulations. However, favoring speed over quality, `rand()` usually produces a short random sequence period which leads to a realistic possibility of random sequence repetition for simulations that require huge amounts of random numbers. Thus StochKit uses the Scalable Parallel Random Number Generators Library (SPRNG),[32,33] which provides multiple high-quality pseudorandom uniform number generators. In addition, SPRNG provides a facility for generating uncorrelated random numbers in parallel. Non-uniform distributions are generated by *ranlib.c*,[34] which provides generators for a wide variety of distributions. Nevertheless, we adapted *ranlib.c* to use SPRNG's linear congruential generator as its uniform generator, to further minimize the probability of sequence repetition. A simple `C++` wrapper was provided for the relevant *ranlib.c* routines to improve user-friendliness.

3.3 Useful Tools

In addition to the core simulation package, StochKit provides three useful tools to support stochastic simulation and analysis. These tools include a simple converter to translate an SBML model file to the input files required by StochKit, a data analyzer to calculate and compare the statistical information from simulation results, and a convenient MPI interface which enables the Monte Carlo simulation ensemble to run on a parallel cluster.

SBML2StochKit Converter SBML (System Biology Markup Language)[35] is a computer-readable format for representing models of biochemical reaction networks. Many biochemical models have been represented with SBML files. For the convenience of SBML users,

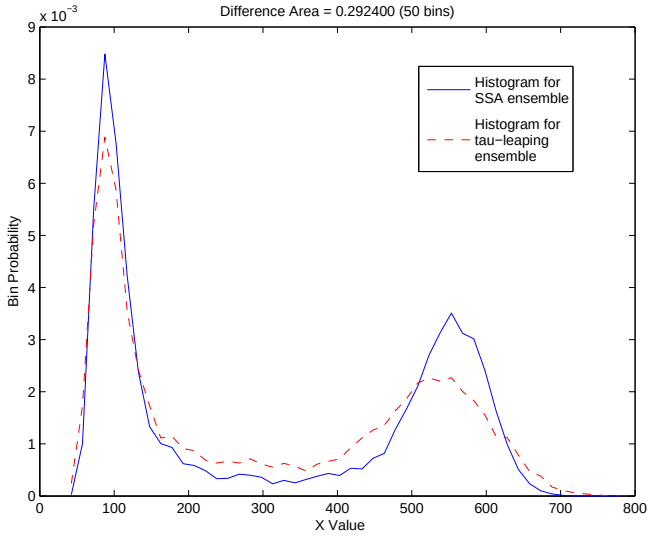

Figure 2: A histogram distance plot for the Schlögl model.[37,38] This plot is based on 10,000 samples of the state variable generated from the SSA and the explicit tau-leaping method with tau = 0.4.

StochKit provides this tool to convert an SBML[35] file to the input files required by StochKit. With this converter, users can conveniently construct their problem files using a separate SBML model builder, make the conversion and run the simulation using StochKit. The current version of StochKit supplies only a command-line implementation for the translation. A more user-friendly GUI is under development.

DataAnalyzer For stochastic modeling, it is important to collect the statistical information from an ensemble of many independent simulations. The DataAnalyzer is a simple Matlab package to generate and plot statistical information from an ensemble. Moreover, the DataAnalyzer provides functions to evaluate the distribution differences[36] between multiple ensembles. For example, the accuracy of different algorithms with different parameters can be measured by calculating the distribution distance[36] between the probability density functions (PDFs) generated via the ensemble from the simulation and the corresponding PDFs from the experimental data, or from an ensemble of an "exact" simulation such as SSA. Figure 2 shows a plot generated by DataAnalyzer, which gives the histogram distance between ensembles of the state variable generated from the SSA and the explicit tau-leaping method with fixed tau value $\tau = 0.4$ for the Schlögl model.[37,38] [1]

MPI Parallel Toolbox In many applications, one has to run a large number of stochastic realizations to collect the ensemble and study the statistics. This task is naturally suited to parallel computation. StochKit provides an MPI toolbox for running many simulations on multiple processors using MPI protocol. We recommend using SPRNG to generate the random numbers since it provides better performance and accuracy, which is particularly

[1]The Schlögl model is a famous chemical reaction system that for certain parameters exhibits bistable behavior in the state variable. Details are available in the references.[37,38]

important in parallel random number generation on a parallel machine.

4 Success Stories

Although the development of StochKit is still in its early stages, there are already around 100 StochKit users worldwide, including model developers who use StochKit to run stochastic simulation and algorithm developers who use and modify StochKit to conduct research on stochastic simulation algorithms. Success stories come from both sides.

Most of the initial success stories come from the side of algorithm development. StochKit provides a simulation framework and a unified interface to help stochastic simulation algorithm research. The modular feature of this package makes the algorithm extension very convenient. For example, if a new stepsize selection formula or even a new simulation formula is to be developed, one only needs to replace the corresponding module in StochKit with the newly developed one and make full use of other modules that are already in the package to generate the simulation and make the comparison. In this way, we have successfully developed the adaptive tau-leaping[22,23] algorithm using StochKit. The convergence[39] and stability[40] of tau-leaping methods have also been studied with the help from StochKit. The convenience of extension and the ability to use parallel computation to generate a large number of independent simulations makes the research cycle dramatically shorter. Besides our own algorithm development, success stories also come from other research groups. Gunawan et al[41] has used StochKit to generate ensembles with different parameters and conducted the parametric sensitivity analysis. Kim et al[42] has used StochKit to speed up their research on the spectral method for sensitivity analysis. Munsky and Khammash[43] have developed a numerical algorithm to approximately solve the chemical master equation by comparing ensemble data generated from StochKit.

On the other side, model developers have made use of this powerful simulation and analysis package to speed up their model development. The most successful application is in the model development of the gene regulatory networks in the heat-shock response (HSR) of E. Coli.[44,45] This model exhibits a multiscale and stochastic nature, which makes the system very stiff. By using multiscale SSA[16] in StochKit, the numerical simulation of the HSR model is 100 times faster than the direct SSA. The time savings in the numerical simulation helped shorten the total time of model development. Another ongoing example is the stochastic cell cycle model of budding yeast[46] by Tyson's research group. They have successfully combined the simulation power of StochKit with the convenient GUI from their model development tool JigCell.[47] With the help of StochKit they have been able to model, simulate and compare the statistics given by the model and the experimental data.[48]

5 Acknowledgement

We would like to thank Andrew Hall, Sotiria Lampoudi and the rest of the StochKit team for their contributions to the development of the StochKit package.

References

[1] D.T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Phys.*, 22:403–434, 1976.

[2] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81:2340–2361, 1977.

[3] H.H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proc. Natl. Acad. Sci. USA*, 94:814–819, 1997.

[4] A. Arkin, J. Ross, and H.H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected E. Coli cells. *Genetics*, 149:1633–1648, Aug 1998.

[5] J. Bruck M. Gibson. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem.*, 105:1876–1889, 2000.

[6] H. Li Y. Cao and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *J. Phys. Chem.*, 121(9):4059–4067, 2004.

[7] J. M. McColluma, G. D. Peterson, C. D. Coxc, M. L. Simpson and N. F. Samatova. The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. *J. Comput. Biol. Chem.*, 30:39–49, Feb. 2005.

[8] I. Beichl J. Blue and F. Sullivan. Faster Monte Carlo simulations. *Physical Rev. E*, 51:867–868, 1995.

[9] T. P. Schulze. Kinetic Monte Carlo simulations with minimal searching. *Physical Review E*, 65(3):036704, 2002.

[10] H. Li and L. Petzold. Logarithmic direct method for discrete stochastic simulation of chemically reacting systems. Technical report, Department of Computer Science, University of California, Santa Barbara, 2006. `http://www.engr.ucsb.edu/~cse`.

[11] StochKit Team. User's guide for stochkit. `http://www.engr.ucsb.edu/~cse`.

[12] M. Yoshimi, Y. Osana, Y. lwaoka, A. Funahashi, N-Hiroi, Y. Shibata, N. lwanaga, H. Kitano and H. Amano. The design of scalable stochastic biochemical simulator on FPGA. In *Proc. of I. C. on Field Programmable Technologies (FPT2005)*, pages 139–140, 2005.

[13] D.T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 115(4):1716–1733, 2001.

[14] C. Rao and A. Arkin. Stochastic chemical kinetics and the quasi steady-state assumption: application to the Gillespie algorithm. *J. Chem. Phys.*, 118:4999–5010, 2003.

[15] D. Gillespie Y. Cao and L. Petzold. The slow-scale stochastic simulation algorithm. *J. Chem. Phys.*, 122:014116, 2005.

[16] D. Gillespie Y. Cao and L. Petzold. Multiscale stochastic simulation algorithm with stochastic partial equilibrium assumption for chemically reacting systems. *J. Comp. Phys.*, 206(2):395–411, 2005.

[17] M. Frankowicz, M. Moreau, P.P. Szczęsny, J. Tóth, and L. Vicente. Fast variables elimination in stochastic kinetics. *J. Phys. Chem.*, 97:1891–1895, 1993.

[18] M. Rathinam, Y. Cao, L.R. Petzold, and D.T. Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *Journal of Chemical Physics*, 2003.

[19] Y. Cao and L. Petzold. Trapezoidal tau-leaping formula for the stochastic simulation of biochemical systems. In *Proceedings of Foundations of Systems Biology in Engineering*, pages 149–152, FOSBE 2005.

[20] E.L. Haseltine and J.B. Rawlings. Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics. *J. Chem. Phys.*, 117(15):6959–6969, 2002.

[21] D.T. Gillespie. The chemical Langevin equation. *J. Chem. Phys.*, 113:297–306, 2000.

[22] D. Gillespie Y. Cao and L. Petzold. Efficient stepsize selection for the tau-leaping method. *J. Chem. Phys.*, 124:044109, 2006.

[23] D. Gillespie Y. Cao and L. Petzold. Avoiding negative populations in explicit tau leaping. *J. Chem. Phys.*, 123:054104–054112, 2005.

[24] D. Gillespie Y. Cao and L. Petzold. The adaptive explicit-implicit tau-leaping method with automatic tau selection. *submitted to J. Chem. Phys.*, 2006.

[25] J. Goutsias. Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems. *J. Chem. Phys.*, 122(18):184102, 2005.

[26] A. Samant and D. G. Vlachos. Overcoming stiffness in stochastic simulation stemming from partial equilibrium: A multiscale Monte Carlo algorithm. *J. Chem. Phys.*, 123, 2005.

[27] D. T. Gillespie Y. Cao and L. R. Petzold. Accelerated stochastic simulation of the stiff enzyme-substrate reaction. *J. Chem. Phys.*, 123, 2005.

[28] E. L. Haseltine and J. B. Rawlings. On the origins of approximations for stochastic chemical kinetics. *J. Chem. Phys.*, 123, 2005.

[29] D. Liu W. E and E. Vanden-Eijnden. Nested stochastic simulation algorithm for chemical kinetic systems with disparate rates. *J. Chem. Phys.*, 123, 2005.

[30] Y. Kaznessis H. Salis. An equation-free probabilistic steady-state approximation: dynamic application to the stochastic simulation of biochemical reaction networks. *J. Chem. Phys.*, 123(21):214106, 2005.

[31] Y. Cao D. Gillespie, L. Petzold. Comment on nested stochastic simulation algorithm for chemical kinetic systems with disparate rates[J. Chem. Phys. 123, 194107 (2005)]. *J. Chem. Phys.*, 126:137101, 2007.

[32] M. Mascagni. SPRNG: A scalable library for pseudorandom number generation. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, San Antonio, Texas, 1999.

[33] M. Mascagni and A. Srinivasan. SPRNG: A scalable library for pseudorandom number generation. In *ACM Transactions on Mathematical Software*, volume 26, pages 436–461, 2000.

[34] B.W. Brown, J. Lovato, and K. Russell. RANLIB.C *Library of C Routines for Random Number Generation*. M.D. Anderson Cancer Center, The University of Texas, 1991.

[35] M. Hucka, A. Finney, H.M. Sauro, H. Bolouri, J. C. Doyle, and H. Kitano. The systems biology markup Language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[36] Y. Cao and L. Petzold. Accuracy limitations and the measurement of errors in the stochastic simulation of chemically reacting systems. *J. Comput. Phys.*, 212:6–24, 2006.

[37] F. Schlögl. On thermodynamics near a steady state. *Zeitschirft fur Physik*, 248:446–58, 1971.

[38] D. Gillespie. *Markov Processes: An Introduction for Physical Scientists*. Academic Press, 1992.

[39] Y. Cao M. Rathinam, L. Petzold and D. Gillespie. Consistency and stability of tau leaping schemes for chemical reaction systems. In *SIAM Multiscale Modeling*, volume 4, pages 867–895, 2005.

[40] M. Rathinam Y. Cao, L. Petzold and D. Gillespie. The numerical stability of leaping methods for stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 121(24):12169–12178, 2004.

[41] L. Petzold R. Gunawan, Y. Cao and F. J. Doyle III*. Sensitivity analysis of discrete stochastic systems. *J. Biophys.*, 88:2530–2540, 2005.

[42] B. J. Debusschere D. Kim and H. N. Najm. Spectral methods for parametric sensitivity in stochastic dynamical systems. *J. Biophys.*, 92:379–393, 2007.

[43] M. Khammash B. Munsky. The finite state projection algorithm for the solution of the chemical master equation. *J. Chem. Phys.*, 124(4):044104, 2006.

[44] H. Kurata, H. El-Samad, T. Yi, M. Khammash and J. Doyle. Feedback regulation of the heat shock response in E. Coli. In *Proceedings of the 40th IEEE conference on Decision and Control*, volume 1, pages 837–842, 2001.

[45] M. Khammash H. Kurata and J. Doyle. Stochastic analysis of the heat shock response in E. Coli. In *3rd International Conference on Systems Biology*, 2002.

[46] J.J. Tyson and B. Novak. Regulation of the eukaryotic cell cycle: Molecular antagonism, hysteresis, and irreversible transitions. *J. Theoretical Biology*, 2001.

[47] Vass, M., N Allen, C.A. Shaffer, N. Ramakrishnan, L.T. Watson, and J.J. Tyson. The jigcell model builder and run manager. *Bioinformatics*, 18:3680–3681, 2004.

[48] J.J. Tyson. Personal communication.