Algorithms and Software for Stochastic Simulation of Biochemical Reacting Systems [*]

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

July 27, 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] The 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 the SSA can be prohibitively high. This is due to the fact that the 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 U. S. Department of Energy under DOE award No. DE-FG03-00ER25430, by the NIH under awards GM078993, GM078992 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

In this paper we review some of the recent work on algorithms for discrete stochastic and multiscale simulation of biochemical reaction networks, and introduce the StochKit software toolkit. In Section 2 we review the basic SSA algorithm, along with the recent efforts to speed up the SSA by reformulation or through the use of parallel computation or special-purpose hardware. Section 3 is devoted to approximate methods for accelerated discrete stochastic and multiscale computation. In Section 4 we describe StochKit and give several examples of its use. Section 5 concludes with some StochKit success stories and a brief discussion of future plans.

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 $[t, t+dt)$.

The original, *direct method* (DM) SSA[1,2] proceeds as follows: We draw two random numbers r_1 and r_2 from the uniform distribution in the unit-interval and take

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

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

where τ is the time to the next reaction, and j is the index of the next reaction. The system is advanced to the next reaction by replacing $t \leftarrow t+\tau$ and $\mathbf{x} \leftarrow \mathbf{x} + \boldsymbol{\nu}_j$.

It is very important to have a fast SSA implementation. This algorithm is likely to be a part of any accelerated or 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] and Logarithmic Direct Method (LDM)[9] with sparse matrix state update are mathematically equivalent formulations which offer improved performance over the original SSA. Often the 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.[10] Some of our current work[11] and others[12] explores the use of novel computer architectures to speed up SSA simulation. However, there is a limit to how much the 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.

3 Accelerated Discrete Stochastic and Multiscale Simulation

Significant speedups of the SSA will inevitably involve *approximations* of one kind or another. In this section we briefly outline some of the most promising methods which employ approximation.

3.1 Tau-Leaping

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.[14] 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 update formula for tau-leaping:

$$\mathbf{X}(t+\tau) \doteq \mathbf{x} + \sum_{j=1}^{M} \mathcal{P}_j\left(a_j(\mathbf{x})\tau\right)\boldsymbol{\nu}_j. \tag{2}$$

Practical considerations cause the actual implementation of tau-leaping to be more complicated than simply substituting into formula (2) 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[15] efficiently estimates the largest τ for which the expected fractional change in every propensity function will be bounded by a user-specified tolerance ϵ (typically $\epsilon = 0.04$). Second is the problem of ensuring that no reactant population be driven negative in a leap. One resolution of this problem[16] is to classify as "critical" all reactions that are within a user-specified number n_c of 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[16] has the additional advantage that, as all reactions become critical, or equivalently as $n_c \to \infty$, it segues smoothly to the exact SSA.

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 *tau* must be chosen based on the fastest timescale, in order to maintain stability of the explicit difference formula. The *implicit tau-leaping* method[17] is a generalization of the implicit Euler method for handling stiff ordinary differential equations, and solves the stability problem. However, it overdamps the natural fluctuations of the *fast* species, and requires that a strategy called "down-shifting"[17] be applied to recover those fast species fluctuations whenever required. There is also a *trapezoidal tau-leaping method*,[18] which 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,[19] which automatically identifies when stiffness is present and dynamically chooses between the explicit and implicit tau-leaping schemes.

3.2 Hybrid Methods

Hybrid methods[20–24] constitute another means of accelerated, approximate simulation for multiscale chemically reacting systems. Hybrid methods combine the traditional deterministic reaction rate equations (RRE) with the SSA. The idea is to split the system into two regimes: the continuous regime and the discrete regime. The RRE are used to simulate the fast reactions between species with large populations. SSA is used for slow reactions or species with small populations.

Hybrid methods efficiently utilize the multiscale properties of the problem. However, there are still some unsolved problems. These methods lack a rigorous theoretical foundation, and there is an intrinsic limitation: when a reaction is fast but one of the corresponding reactants has a small population, it should not be placed in the deterministic regime. Thus this reaction must still be handled by the SSA, resulting in a very slow simulation. In fact, the tau-leaping methods cannot handle this situation either, because the propensities can change very rapidly.

3.3 Slow-Scale SSA Method

Another approach to handling stiff systems, which can effectively handle the situation of a fast reaction involving species that are present with very small population, involves a stochastic generalization of the quasi-steady state and partial (or rapid) equilibrium methods of deterministic chemical kinetics.[22,25–34] The theoretical basis for these methods is captured most comprehensively by the *slow-scale SSA*[26,30,34] and its practical implementation as the *multiscale SSA*.[28] The slow-scale SSA is a systematic procedure for partitioning the system into fast and slow reactions, and then simulating *only the slow reactions*, using specially modified propensity functions. 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 Michaelis-Menten enzyme-substrate reaction typically falls within the provenance of the slow-scale SSA.[30]

4 StochKit– Stochastic Simulation Toolkit

4.1 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. There are several published software packages[35–38] available for stochastic simulation of biochemical

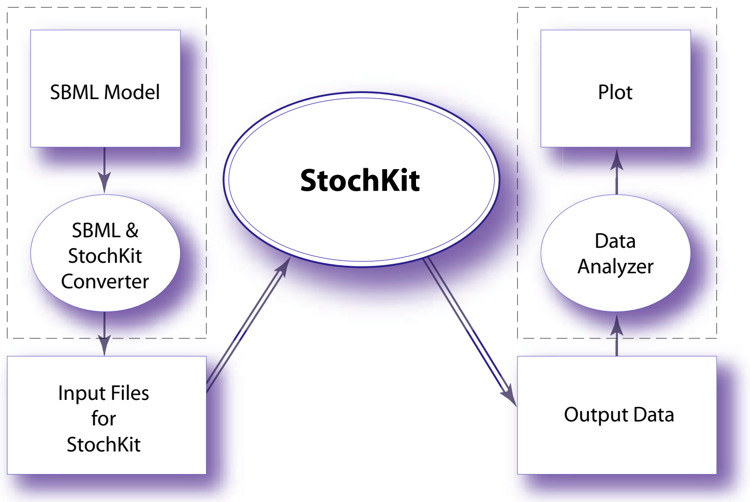

Figure 1: Simulation Process of StochKit

networks. They provide nice graphic interfaces with some stochastic or hybrid simulation tools. The goal of StochKit is to provide a state of the art and extendable stochastic simulation platform with the most efficient implementation, based on rigorous theoretical foundations.

StochKit consists of a suite of software for stochastic simulation. The StochKit core implements the simulation algorithms. Additional tools are provided for convenience. A typical simulation process of StochKit is shown in Figure 1. If an SBML model file is available, the SBML2StochKit Converter can retrieve the model information and write into a StochKit input file. Otherwise the user needs to provide the model properties required in the input file of StochKit . With the input file and the provided driver file, StochKit simulates the model and puts the results into a file. Then the output data can be analyzed with Data Analyzer or other tools.

The structure of StochKit is shown in Figure 2. where the **Math** directory contains the library of linear algebra and random number generators, the **StochRxn** directory contains the core functions of StochKit, the **Test** directory contains a few examples which the users can use as a template for their own application, and the **Tools** directory contains three useful tools for data analysis as explained later in this section.

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.

4.2 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

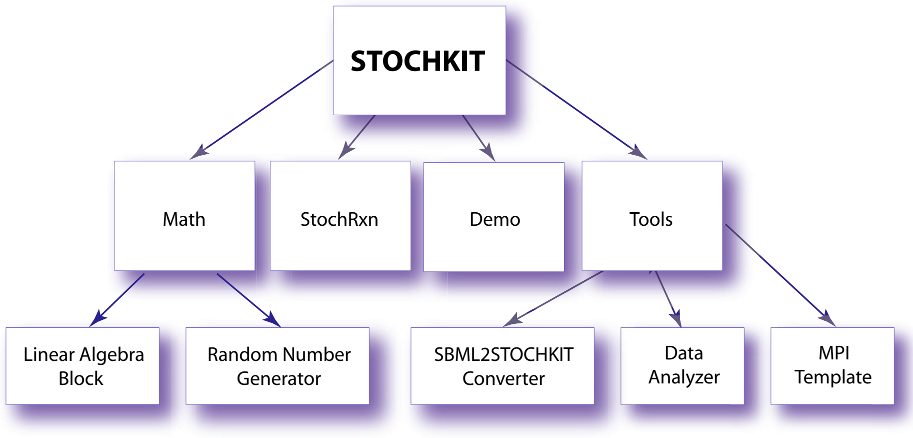

Figure 2: Structure of StochKit

algorithm,[6] tau-leaping methods (explicit tau-leaping,[13] implicit tau-leaping[17] and trapezoidal tau-leaping[18] with fixed stepsize; and adaptive stepsize, nonnegativity-preserving explicit tau-leaping method[15,16]), slow scale SSA[26] and multiscale SSA[28] methods. These algorithms were implemented as modules. Users who are not experts in stochastic simulation can simply use the default option to let StochKit automatically select the simulation algorithm and the corresponding stepsize. Advanced users can select different options for better performance or accuracy (a general guide on how to choose these algorithms is given in Table 1). Those developing new algorithms (or simply refining existing ones) need only

Population [1]	*Reaction Scale* [2]	*Applicable Methods*	*Suggested Options*
Small $(0 \sim 10)$	*nonstiff*	*SSA, ET, AT*	*AT*
	stiff	*SSA, ssSSA*	*ssSSA*
Medium $(10 \sim 1,000)$	*nonstiff*	*SSA, ET, AT*	*AT*
	stiff	*SSA, IT, TT, ssSSA*	*ssSSA*
Large $(> 1,000)$		*tau-leaping methods* *SDE or ODE methods*	*AT*

Table 1: A general guide for applicability of different methods, where ET represents explicit tau-leaping, AT represents adaptive explicit tau-leaping, IT represents implicit tau-leaping, TT represents trapezoidal tau-leaping and ssSSA represents slow-scale SSA. The suggested options are our recommendations, from the methods available in StochKit.

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

[1]For the purpose of this table, 'population' should be considered large if the population of *every* species in the system is large. Otherwise, it should be considered small.

[2]Roughly speaking, a system is stiff if it involves some reactions with slow rates, and other reactions with every fast rates.

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 (For details, see the User's Guide[10]). The capacity to run an ensemble of stochastic simulations is provided, with an option to compute in parallel via MPI.[39]

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. 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),[40,41] 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*,[42] 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.

4.3 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)[43] 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, StochKit provides this tool to convert an SBML[43] 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.

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[44] between multiple ensembles. For example, the accuracy of different algorithms with different parameters can be measured by calculating the distribution distance[44] 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.

MPI Parallel Toolbox In many applications, one must 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 important in parallel random number generation.

4.4 Simulation Examples

Two simple models are chosen to illustrate the application of StochKit. One is the simple decaying dimerization model.[13] The other is the Schlögl model.[45] The simulations were done on a personal computer with Intel(R) Pentium(R) 4 2.80GHz CPU.

1. Decaying Dimerization Model

This model is given by

$$\begin{aligned} S_1 &\xrightarrow{c_1} 0 \\ S_1 + S_1 &\underset{c_2}{\overset{c_3}{\rightleftharpoons}} S_2 \\ S_2 &\xrightarrow{c_4} S_3. \end{aligned} \tag{3}$$

We choose different initial conditions and reaction rates to show different situations.

- Case 1: Initial conditions are $X_1 = 10^5,\ X_2 = X_3 = 0$, and $FinalTime = 10$. The reaction rate constants are $c_1 = 1,\ c_2 = 0.002,\ c_3 = 0.5,\ c_4 = 0.04$.
- Case 2 (stiff): Initial conditions are $X1 = 400,\ X2 = 798,\ X3 = 0$, and $FinalTime = 0.2$. The reaction rates are $c_1 = 1$, $c_2 = 10$, $c_3 = 1000$, $c4 = 0.1$.

For the two cases we collected 10,000 samples with the SSA (Direct Method), the adaptive explicit tau-leaping method, and the slow-scale SSA (when applicable). The computational cost and accuracy comparison results are shown in Table 2, where the accuracy was measured by the distribution distance[44] between the ensembles of the SSA and the corresponding simulation method. The accuracy given for the SSA is the 'self-distance', which is the distance between 10,000 SSA runs and 10,000 SSA runs using a different sequence of random number. From the simulation results we can see that the adaptive explicit tau-leaping method is the best option for case 1. In case 2, the slow-scale SSA is the best option because the model is stiff.

Model	*Method*	*SSA*	*ATau*	*ssSSA*
Case 1	*CPU time*	*145.2*	*28.0*	*NA*
	Distribution Error	*0.107*	*0.12*	*NA*
Case 2	*CPU time*	*1753.7*	*404.2*	*0.96*
	Distribution Error	*0.0682*	*0.12*	*0.12*

Table 2: Computational cost and accuracy comparison of different methods for the decaying dimerization model.

2. Schlögl Model

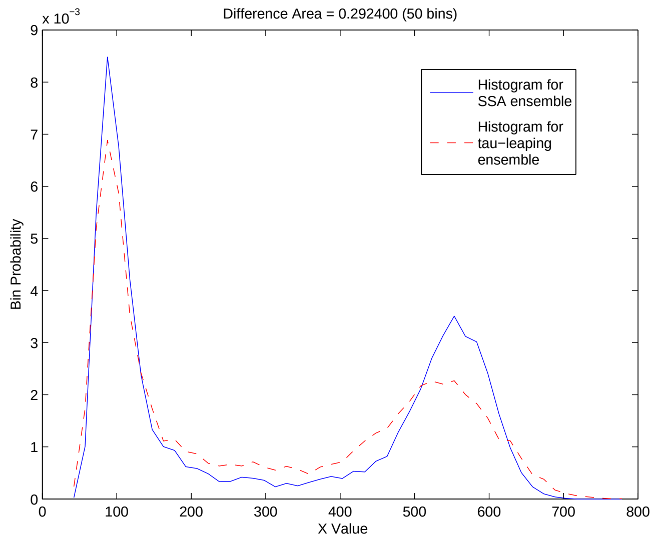

Figure 3: A histogram distance plot for the Schlögl model.[45,46] This plot is based on 10,000 samples of the state variable generated from the SSA and the explicit tau-leaping method with tau = 0.4.

This model is given by

$$\begin{aligned} B_1 + 2X &\underset{c_2}{\overset{c_1}{\rightleftharpoons}} 3X, \\ B_2 &\underset{c_4}{\overset{c_3}{\rightleftharpoons}} X, \end{aligned} \tag{4}$$

where B_1 and B_2 denote buffered species whose molecular populations N_1 and N_2 are assumed to remain essentially constant over the time interval of interest. We used the parameter $c_1 = 3 \times 10^{-7},\ c_2 = 10^{-4},\ c_3 = 10^{-3},\ c_4 = 3.5;\ N_1 = 1 \times 10^5,\ N_2 = 2 \times 10^5$.

The Schlögl model is famous for its bi-stability, shown in Figure 3. If the reactions are simulated deterministically, the solution goes to one side of the distribution or the other, depending on the initial case. A stochastic model is required to capture the full range of behavior of the system.

The simulation results of 10^6 samples generated by different methods are shown in Table 3. We can see that adaptive explicit tau-leaping method is the best option. Because the system is not stiff, we did not consider the ssSSA.

Method	*SSA*	*ATau*
CPU time	*7239.3*	*268.8*
Distribution Error	*0.02478*	*0.041*

Table 3: Computational cost and accuracy comparison of different methods for the Schlögl model.

5 Success Stories and Future Plans

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[15,16] algorithm using StochKit. The convergence[47] and stability[48] of tau-leaping methods have also been studied with 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[49] has used StochKit to generate ensembles with different parameters and conducted the parametric sensitivity analysis. Kim et al[50] has used StochKit to speed up their research on the spectral method for sensitivity analysis.

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.[51,52] This model exhibits a multiscale and stochastic nature, which makes the system very stiff. By using multiscale SSA (which is closely related to ssSSA)[28] in StochKit, the numerical simulation of the HSR model is 100 times faster than with the direct SSA. The time savings in the numerical simulation helped to shorten the total time of model development. Another ongoing example is the stochastic cell cycle model of budding yeast[53] by Tyson's research group. They have successfully combined the simulation power of StochKit with the convenient GUI from their model development tool JigCell.[54] With the help of StochKit they have been able to model, simulate and compare the statistics given by the model and the experimental data.[55]

We continue to develop new algorithms and add them to StochKit. An SSA implementation based on the Logarithmic Direct Method[9] will be added very soon, and we are working on implementations of the adaptive explicit-implicit tau-leaping method[19] and the slow scale tau-leaping method[56] for StochKit. We are working with the CalTech SBML group on an ambitious, user-friendly software package that will include the simulation capabilities of StochKit. We are experimenting with spacial-purpose hardware such as Graphics Processing Units (GPU),[11] as well as parallel and grid computing to enable discrete stochastic simulation for larger and more complicated models.

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

[6] Y. Cao, H. Li 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. Cox, 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] J. Blue, I. Beichl and F. Sullivan. Faster Monte Carlo simulations. *Physical Rev. E*, 51:867–868, 1995.

[9] 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.

[10] StochKit Team. User's Guide for StochKit. http://www.engr.ucsb.edu/~cse.

[11] H. Li and L. Petzold. Efficient parallelization of stochastic simulation algorithm for chemically reacting systems on the graphics processing unit. In preparation.

[12] M. Yoshimi, Y. Osana, Y. lwaoka, A. Funahashi, N-Hiroi, Y. Shibata, N. lwanaga, H. Kitano and H. Amano. The design of a 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] D.T. Gillespie. The chemical Langevin equation. *J. Chem. Phys.*, 113:297–306, 2000.

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

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

[17] 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.

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

[19] Y. Cao, D. Gillespie and L. Petzold. The adaptive explicit-implicit tau-leaping method with automatic tau selection. *J. Chem. Phys.*, 126(22):224101–224101–9, 2007.

[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] H. Salis and Y. Kaznessis. Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *J. Chem. Phys.*, 122(5):54103, 2005.

[22] D. Adalsteinsson R. Erban, I. Kevrekidis and T. Elston. Gene regulatory networks: a coarse-grained, equation-free approach to multiscale computation. *J. Chem. Phys.*, 124(17):084106, 2006.

[23] A. Chatterjee and D. G. Vlachos. Multiscale spatial Monte Carlo simulations: Multigridding, computational singular perturbation, and hierarchical stochastic closures. *J. Chem. Phys.*, 124(6):64110, 2006.

[24] Y. Kaznessis. Multi-scale models for gene networks. *Chem. Eng. Sci.*, 61(3):940, 2006.

[25] 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.

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

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

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

[29] 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.

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

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

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

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

[34] D. Gillespie, L. Petzold and Y. Cao. 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.

[35] A. Ribeiro and J. Lloyd-Price. SGN Sim, a Stochastic Genetic Networks Simulator. *Bioinformatics*, 23:777–779, 2007.

[36] S. Ramsey, D. Orrell, H. Bolouri. Dizzy: Stochastic simulation of large-scale genetic regulatory networks. *J. Bioinformatics Comput. Biol.*, 3:415–436, 2005.

[37] H. Salis, V. Sotiropoulos and Y. Kaznessis. Multiscale Hy3S: Hybrid stochastic simulation for supercomputers. *BMC Bioinformatics*, 2006, 7:93.

[38] D. Adalsteinsson, D. McMillen and T. Elston. Biochemical Network Stochastic Simulator (BioNetS): software for stochastic modeling of biochemical networks. *BMC Bioinformatics*, 2004, 5:24.

[39] Message Passing Interface (MPI) Tutorial. (http://www.llnl.gov/computing/ tutorials/mpi/.

[40] 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.

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

[42] 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.

[43] 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.

[44] 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.

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

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

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

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

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

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

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

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

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

[54] 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.

[55] J.J. Tyson. Personal communication.

[56] Y. Cao and L. Petzold. Slow scale tau-leaping method. Submitted.