We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Efficient Stochastic Simulation to Analyze Targeted Properties of Biological Systems

Hiroyuki Kuwahara[1], Curtis Madsen[2], Ivan Mura[3], Chris Myers[2], Abiezer Tejeda[4] and Chris Winstead[4]
[1]*Carnegie Mellon University, U.S.A.*
[2]*University of Utah, U.S.A.*
[3]*Microsoft Research - University of Trento CoSBi, Italy*
[4]*Utah State University, U.S.A.*

1. Introduction

Randomness in gene expression has been ubiquitously observed from primitive prokaryotes to higher eukaryotes (Elowitz et al., 2002; Johnston & Desplan, 2010; Losick & Desplan, 2008; Maheshri & O'Shea, 2007; Raj & van Oudenaarden, 2008; Raser & O'Shea, 2004; Wernet et al., 2006). Yet, it is widely recognized that living organisms have evolved to control and exploit such underlying stochastic noise to optimize their dynamic characteristics to better cope with and compete in their living environments in a variety of ways (Arkin & Fletcher, 2006). For example, on the one hand, a stochastic switch has been shown to probabilistically regulate expression of an adhesive virulence factor in the main causative pathogen of uncomplicated lower urinary tract infections based on ambient temperature (Gally et al., 1993). On the other hand, marine embryo development seems to work much more deterministically and reliably at different rates over a range of environmental conditions (Istrail et al., 2007). Gaining insights into how living organisms control stochastic effects to achieve specific functions, thus, can have a significant implication in enhancing many aspects of our lives. That is, for example, by understanding the control mechanisms associated with the etiology and stability of complex non-Mendelian diseases, novel and effective therapies for prevention and treatment of such diseases can be developed. However, owing to sheer-size complexity of even a relatively simple biological system, elucidation of stochastic control mechanisms at the molecular level may not be something which can be efficiently and effectively accomplished with the current limitation of controllability and observability in wet-lab experiments alone. This, in turn, makes computational analysis essential to any efforts aimed at understanding of control and information processing mechanisms of intricate biological systems.

Detailed-level stochastic effects in biological systems are—by and large—captured and analyzed by devising the *stochastic chemical kinetics* (SCK) framework (Samoilov & Arkin, 2006). Assuming that the system is spatially homogeneous, the SCK model specifies the time-homogeneous probabilistic reaction rate function of each reaction and discrete changes in the molecular species populations through individual discrete reaction events. While sample trajectories of a SCK model can be accurately realized via Gillespie's *stochastic simula-*

tion algorithm (SSA) (Gillespie, 1976; 1977), the computational requirements of the SSA can be substantial due largely to the fact that it not only requires a potentially large number of simulation runs in order to estimate the system behavior at a reasonable degree of statistical confidence, but it also requires every single elementary-reaction event to be simulated one at a time. As recent advances in experimental techniques have enabled us to unveil more key components and more detailed organization structures of many biological systems, and as we are beginning to address more complex and sophisticated biological questions than ever before, it has become increasingly clear that no single modeling and simulation method can satisfy the needs of a wide spectrum of such complex questions.
One approach to alleviate the computational requirements involved in analysis of SCK models is to speed up the simulation of individual SSA by letting go of exactness. An example of this is τ-leaping method (Gillespie, 2001), which approximates the number of firings of each reaction in a predefined interval rather than executing each reaction individually. Another example is model reduction, which abstracts away dynamically insignificant reactions or species in order to make the overall systems biology analysis more efficient (Kuwahara et al., 2010; 2006).
Another approach to accelerate the analysis of SCK model is to tailor stochastic simulations based on specific dynamical properties of interest and apply a more suitable simulation method than the standard SSA. This chapter describes two such approaches to efficiently analyze various dynamical properties of interest. The rest of this chapter is organized as follows. Section 2 briefly describes SCK and SSA. Section 3 presents a modified SSA to better quantify the normal or typical behavior. Section 4 presents another modified SSA for the analysis of rare deviant events. Section 5 presents a case study analysis of enzymatic futile cycles. Finally, Section 6 presents our conclusions.

2. Stochastic Chemical Kinetics

Stochastic chemical kinetics (SCK) is a theoretical framework that accounts for the statistics of randomly-occurring chemical reactions (Gillespie, 1976; 1977; 2005; 2007). In the SCK framework, a reaction system consists of a liquid volume, Ω, containing a population of randomly-moving molecules. The molecules represent one or more species types. The medium is typically assumed to be "well-stirred," meaning a given reactant molecule may be found at any position in the medium, and may be moving in any direction, with uniform probability. A reaction may occur whenever there is a collision among the respective reactant molecules. When a reaction occurs, the reactants are removed from Ω, and the products are added to Ω.
Under the SCK framework, a reaction system's time-evolution is governed by a set of probability laws which can be deduced through combinatorial methods. Suppose a reaction system consists of N chemical species s_i, and the volume Ω contains x_i molecules of species s_i at a specific time t, for $i = 1, ..., N$. Also, suppose the system has M reactions R_1, R_2, ..., R_M, and each reaction R_j has a set of reactants $\mathcal{R}_j$. Finally, let r_{ij} be the number of reactants of species i that participate in reaction R_j, and let p_{ij} be the number of products of species i that are produced by reaction R_j. Let $\boldsymbol{\nu}_j$ be the change in $\mathbf{x}$ that results from the occurrence of R_j. The elements of $\boldsymbol{\nu}_j$ are given by $\nu_{ij} = p_{ij} - r_{ij}$.
Given these definitions, the SSA algorithm computes the following probabilities:

- $a_j(\mathbf{x}, t)\, dt$ = the probability, given state $\mathbf{x}$ at time t, that R_j occurs in a small time-interval of width dt. This is called the *propensity function* of reaction j.

- $P_0(\tau|\mathbf{x}) =$ the probability, given state $\mathbf{x}$ at time t, that *no reaction* occurs in the time-interval $(t, t+\tau)$. It can be shown that $P_0 = \exp\left(-\tau \sum_{j=1}^{M} a_j\right)$, hence P_0 is fully determined by the reactions' propensities.
- The product of these is called the *reaction pdf*, given by $f_R(\tau, j)\, dt = a_j(\mathbf{x}, t)\, P_0(\tau|\mathbf{x})\, dt$. The reaction pdf expresses the probability that the next reaction is R_j, and it occurs at time $t+\tau$.

Stochastic simulation algorithms use the reaction pdf to generate a sequence of reaction events. Because $f_R(\tau, j)$ is fully determined by the propensities, we may fully characterize the system's time-evolution by computing all the a_j terms. In order to compute the a_j terms, Gillespie proposed the *fundamental hypothesis of SCK*:

$$a_j = c_j \times h_j \tag{1}$$

where

$$c_j = \text{the } \textit{stochastic reaction constant}, \tag{2}$$

$$h_j = \text{the total number of combinations of reactants in } \mathcal{R}_j. \tag{3}$$

The stochastic reaction constant c_j is closely related to the traditional reaction-rate constant k_j. It is generally possible to compute c_j from k_j. In many cases, especially with regard to genetic reaction networks, the actual reaction rates are not well known. In these cases, the c_j are estimated by making an educated guess. In some cases, careful experiments have been carried out to determine the reaction constants, but these are in the minority. As a rule of thumb, the c_j constants generally lie between 10^{-4} and 0.1 for "slow" and "fast" reactions, respectively. When the c_j are not precisely known, their estimated values may be tuned within this range to reflect the relative speed expected from each reaction.
The number of reactant combinations, h_j is found by a combinatorial analysis. If reaction R_j involves multiple distinct reactants, as in $s_1 + s_2 \rightarrow s_3$, then the number of reactant combinations is the product over the reactant populations: $h_j = x_1 \times x_2$. If R_j has multiple reactants of the same species, as in $2s_1 \rightarrow s_2$, then the number of combinations is found by the n-choose-k function, in this case $h_j = 0.5x_1(x_1 - 1)$. In general, the total combinations is given by a product over n-choose-k calculations:

$$h_j = \prod_{i \in \mathcal{R}_j} \binom{x_i}{r_{ij}}. \tag{4}$$

For example, the two-reaction system model shown in Figure 1 contains the reaction $s_2 + 2s_3 \rightarrow s_1$. The number of combinations for this reaction equals the number of s_2 molecules, times the number of pairs of s_3 molecules, i.e. $h_2 = x_2 \times 0.5x_3(x_3 - 1)$.

2.1 Gillespie's Stochastic Simulation Algorithm (SSA).

To simulate the time-evolution of a reaction network, one may use the reaction pdf to generate a sequence of random reaction events, starting from a specified initial state $\mathbf{x}(t_0)$ at start-time t_0. The simulation yields a sequence of system states $\mathbf{x}(t_1)$, $\mathbf{x}(t_2)$, ... that occur at non-uniform times t_1, t_2, ..., and so on. This sequence is referred to as a *sample path*, which represents a physically plausible sequence of reactions.
To generate a sample path, we proceed one reaction at a time. For each reaction, we generate two random numbers:

1. $\tau =$ the time to the next reaction, and
2. $R_\mu =$ the reaction that fires at time $t + \tau$.

We assume the system begins in a specified initial state $\mathbf{x}_0$ at start-time t_0. The SSA is executed by repeating three essential tasks: (1) Generate a time for the next reaction to occur, (2) Generate the reaction that occurs at that time, and (3) Change the state $\mathbf{x}$ to reflect that the reaction has occurred. Algorithm 1 implements these tasks with the proper statistics and produces a physically realistic sample path.

$\begin{pmatrix} s_1 + s_2 \rightarrow s_3 \\ s_2 + 2s_3 \rightarrow s_1 \end{pmatrix}$	$\begin{aligned} h_1 &= x_1 \times x_2 \\ h_2 &= x_2 \times 0.5x_3\,(x_3 - 1) \end{aligned}$
(a) Reactions	*(b) Combinations*
$\begin{aligned} \mathbf{r}_1 &= \langle 1, 1, 0 \rangle \\ \mathbf{r}_2 &= \langle 0, 1, 2 \rangle \end{aligned}$	$\begin{aligned} \boldsymbol{\nu}_1 &= \langle -1, 1, 1 \rangle \\ \boldsymbol{\nu}_2 &= \langle 1, -1, -2 \rangle \end{aligned}$
(c) Reactants	*(d) State changes*

Fig. 1. An example of a simple two-reaction system model.

Algorithm 1 Gillespie's stochastic simulation algorithm.

$t \leftarrow 0$
$\mathbf{x} \leftarrow \mathbf{x}_0$.
while $t < t_{max}$ **do**
  for $j = 1$ to M **do**
    evaluate $h_j(\mathbf{x})$ and $a_j(\mathbf{x})$.
  end for
  Calculate a_0
  $r_1 \leftarrow$ a randomly generated number from $\mathcal{U}(0, 1)$.
  $r_2 \leftarrow$ a randomly generated number, uniformly distributed in the open interval $(0, 1)$.
  $\tau \leftarrow \left(\frac{1}{a_0}\right) \ln\left(\frac{1}{r_1}\right)$.
  $\mu \leftarrow$ the least integer for which $r_2 \leq \frac{1}{a_0} \sum_{j=1}^{\mu} a_j$. Then R_μ is the next reaction that occurs at time $t + \tau$.
  $\mathbf{x} \leftarrow \mathbf{x} + \boldsymbol{\nu}_\mu$.
  $t \leftarrow t + \tau$.
end while

2.2 Approximations of the SSA

A number of researchers have devised methods to reduce the computational complexity of stochastic simulations. In most cases, these methods rely on approximations that are valid

under a restricted set of reaction conditions. Two of the best-known SSA approximations were devised by Gillespie, the τ-leaping method and the chemical Langevin equation (CLE) method (Gillespie, 2001). These methods greatly improve the speed of stochastic simulations in many types of reactions. These methods also help to establish the incremental SSA methods described in section 3.

2.2.1 The τ-leap Method

The τ-leap method improves simulation speed by firing many reactions at the same time. By contrast, the original SSA must compute each separate reaction individually. By computing a bundle of reactions simultaneously, the τ-leap method can run many times faster than the ordinary SSA. The τ-leap method requires a *leap condition* which is that all of the propensity functions a_j remain approximately constant during a sufficiently small time-interval τ. Under this approximation, the propensity for a given reaction R_j during the τ-window is assumed independent of other reactions that may occur during the same time-window. Let k_j be the number of times reaction R_j occurs during the time-window. Then k_j can be shown to be a Poisson distributed random variable. A sample-path can therefore be generated using the modified SSA steps shown in Algorithm 2.

Algorithm 2 The τ-leaping method.

1: $t \leftarrow 0$
2: $\mathbf{x} \leftarrow \mathbf{x}_0$.
3: **while** $t < t_{max}$ **do**
4: **for** $j = 1$ to M **do**
5: evaluate $h_j(\mathbf{x})$ and $a_j(\mathbf{x})$.
6: $k_j \leftarrow$ a random number generated from the Poisson distribution $\mathcal{P}\left(a_j\tau\right)$.
7: **end for**
8: $\mathbf{x} \leftarrow \mathbf{x} + \sum_{j=1}^{M} k_j \boldsymbol{\nu}_j$.
9: $t \leftarrow t + \tau$.
10: **end while**

The leap condition is most likely satisfied in systems with large molecule counts. For these systems, a single reaction produces only a small relative change in the system's state. The change in propensities is correspondingly small. There are convenient run-time methods that test the τ-leap conditions, and adaptively determine the optimal time-step (Gillespie, 2001; Gillespie & Petzold, 2003). In the context of genetic circuits, application of the τ-leap method is complicated by the low count of DNA molecules. Reactions including DNA transcription and translation may induce rapid changes in propensities across the reaction system.

2.2.2 The Chemical Langevin Equation Method

The Chemical Langevin Equation (CLE) method is a further approximation to the τ-leap method that applies in systems with very large molecule counts (Gillespie, 2000; 2001). In addition to the Leap Condition, the CLE method requires that $\tau \gg \max_j \left\{1/a_j\right\}$. If these conditions are satisfied, then the discrete Poisson distribution $\mathcal{P}\left(a_j\tau\right)$ approaches the continuous Gaussian (Normal) distribution $\mathcal{N}(\mu, \sigma)$ with $\mu = \sigma^2 = a_j\tau$. The τ-leap method is therefore modified slightly to produce Algorithm 3.

Algorithm 3 The CLE method.

$t \leftarrow 0$
$\mathbf{x} \leftarrow \mathbf{x}_0$.
while $t < t_{max}$ **do**
  for $j = 1$ to M **do**
    evaluate $h_j(\mathbf{x})$ and $a_j(\mathbf{x})$.
    $k_j \leftarrow$ a random number generated from the Gaussian distribution with mean $a_j\tau$ and variance $a_j\tau$.
  end for
  $\mathbf{x} \leftarrow \mathbf{x} + \sum_{j=1}^{M} k_j \boldsymbol{\nu}_j$.
  $t \leftarrow t + \tau$.
end while

The CLE method is applicable in systems where molecule counts are so large that they may be approximated as continuous values. The CLE method provides a segue between stochastic chemical kinetics and traditional deterministic reaction-rate equation (RRE) models. RRE models use continuous-valued ordinary differential equations (ODEs) to model a reaction system's time-evolution. In the limit as all $a_j\tau \to \infty$, the CLE system converges to a deterministic ODE system. Hence the τ-leap and CLE conditions provide insight into the implicit assumptions that underlie the widely used RRE methods. In systems where the τ-leap and CLE conditions are not satisfied (or are only weakly satisfied), continuous ODE models are invalid and a suitable SCK approach should be used.

3. Determining Typical Behavior

In the analysis of stochastic biochemical systems, one starts with two basic questions. First, what is the system's normal or typical behavior? Second, how robust is that behavior? These questions are especially important for custom-designed biochemical networks, such as synthetic genetic circuits. In this case, the designer is interested in verification that the system's actual behavior matches the designer's intent. This section presents the *incremental* SSA (iSSA) which sets out to answer these questions in small time-increments. This approach has some characteristics in common with the popular SPICE program for simulating electronic circuits (Nagel & Pederson, 1973). The main objective of iSSA is to provide a first-step verification solution for synthetic biochemical systems.

Stochastic simulation algorithms generally provide a single snapshot of a system's possible behavior. If the system exhibits a high level of stochastic activity, the underlying behavior may be obscured by transient "noise". In order to understand the range of typical behaviors, many simulation runs are needed. It is common practice to compute a simulation envelope with the form $\bar{\mathbf{x}} \pm \boldsymbol{\sigma}$, where $\bar{\mathbf{x}}$ and $\boldsymbol{\sigma}$ are the average and standard deviation vectors computed over K SSA sample paths. The average, $\bar{\mathbf{x}}$, is considered to be the system's typical behavior. The standard deviation, $\boldsymbol{\sigma}$, indicates the degree to which the system is expected to deviate from the typical behavior.

The direct average method is suitable for systems that are only weakly stochastic. Direct averaging is not suitable in systems that have multiple operating modes, leading to many divergent behaviors that are all "typical." Unfortunately, the majority of interesting biochemical systems, especially genetic circuits, fall into this category. For example, consider a bi-stable

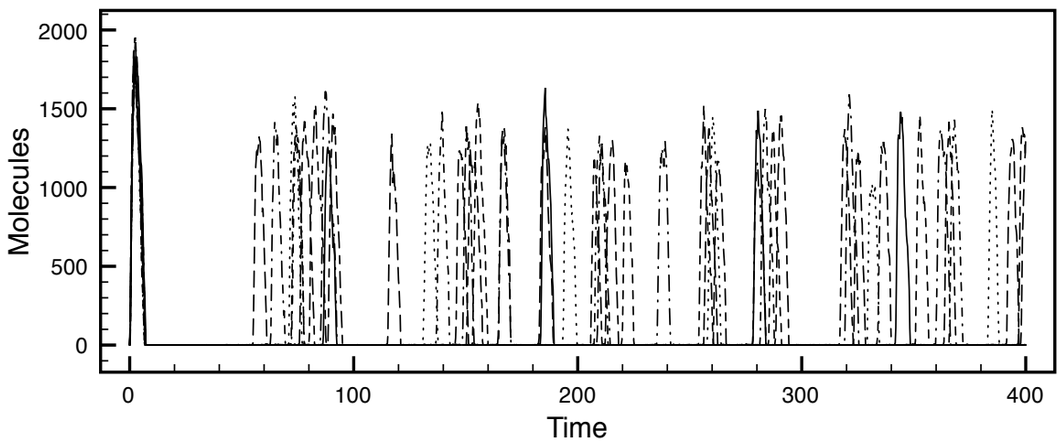
system that randomly converges toward one of two states. Suppose half of all SSA sample-paths arrive at state 1, and the other half arrive at state 2. By averaging over all SSA runs, one obtains a fictitious middle state that obscures the system's true typical behaviors.
The averaging problem is further compounded in dynamic systems that switch between states, particularly if the state-transitions occur at random times. The simplest example of a dynamic multi-state system is a stochastic oscillator in which the production of some signal is alternately activated and inhibited. One such oscillator is the circadian rhythm model developed by Vilar et al. (2002). Stochastic simulation runs of the circadian rhythm are shown in Fig. 2(a), and the average over all runs is shown in Fig. 2(b). When production is activated, it stimulates a brief but intense production of an output molecule A. As the amount of A increases, it represses its own production and eventually degrades back to zero. This pattern creates "pulses" of A that occur at random times. Because the pulse times are random, they generally do not occur at the same times in different simulation runs. If a direct average is computed, the mis-aligned impulses tend to be masked by the averaging (Samad et al., 2005). The circuit's most relevant and interesting behaviors are consequently concealed by the averaging.
In order to obtain meaningful aggregate information from many SSA runs, the iSSA was proposed by Winstead et al. (2010). In the iSSA, a conventional SSA is executed K times over a short time increment. The time increment is chosen such that the circuit's state changes slightly during the increment, similar to the τ-leaping method described in Section 2. Statistics are gathered at the end of the time increment. A new circuit state is selected from those statistics, and the algorithm is repeated for another increment.
By computing average changes over small time increments, iSSA reveals the typical behavior occurring in each increment. The results of iSSA are stochastic, and repeated iSSA simulations may yield different results. For example, iSSA may be used to simulate a bi-stable stochastic system. The iSSA method follows a cluster of sample paths that are close to each other, and hence tends to arrive at one of the two stable states.

3.1 iSSA Overview

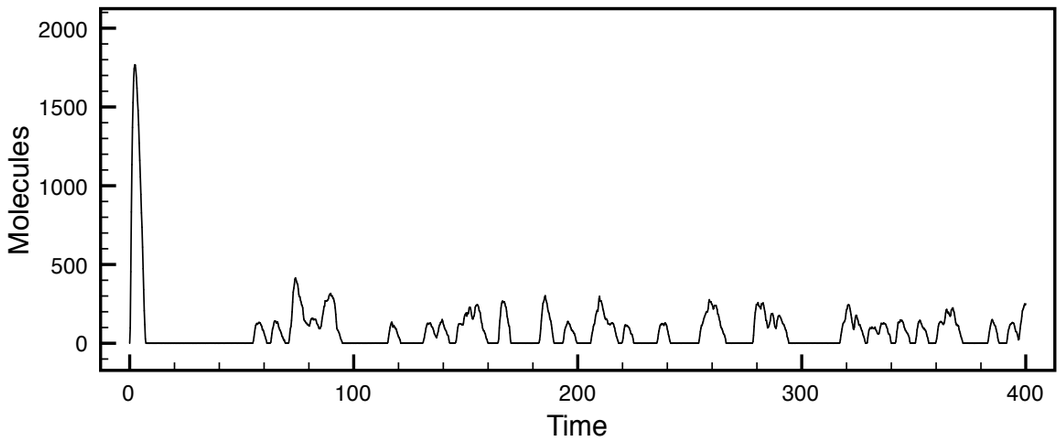
The general steps of the iSSA are shown in Algorithm 4. The iSSA is wrapped around a core SSA algorithm, and may be specialized to perform a variety of incremental analyses. The generic iSSA works by choosing some initial condition for an SSA run, then executing the SSA over a brief interval. Lastly, the iSSA performs some analysis on the incremental SSA results before proceeding to the next increment. The physical interpretation of iSSA results depends on the particular implementation of the `select` function, which define how the SSA simulation conditions are chosen, and the `process` function, which defines how the SSA results are analyzed. This chapter assigns these functions to achieve a *marginal probability density evolution*, and it is known as iSSA-MPDE. The iSSA also allows the use of different SSA methods, such as the τ-leaping or CLE methods, when permitted by the reaction conditions. The following sections examine the derivations and conditions that apply to the iSSA.

3.2 Derivation of iSSA-MPDE

The goal of an iSSA that uses marginal probability density evolution is to provide an alternative approach that reveals the time-evolution of the statistical envelope for each species, under appropriate system conditions. The function definitions for iSSA-MPDE are given in Table 1. In essence, iSSA-MPDE approximates each species as an independent Gaussian-distributed random variable. At the start of each SSA run, the initial molecule counts are randomly generated using each species' marginal Gaussian probability distribution. After all K SSA runs



(a)



(b)

Fig. 2. (a) SSA simulations of a stochastic oscillator. (b) The average response over all SSA sample-paths, revealing incoherent results due to misaligned SSA events.

Algorithm 4 The general iSSA framework.

$t \leftarrow 0$
`initialize` the state-information structure S using initial state $\mathbf{x}_0$.
while $t < t_{max}$ **do**
  for $k = 1$ to K **do**
    `select` a state $\mathbf{x}$ based on the state-information S.
    perform one SSA run with start time t, max-time $t + \tau$ and initial state $\mathbf{x}$.
    `record` the ending SSA state $\mathbf{x}'$ by appending it to a state-table $\mathbf{X}'$.
  end for
  `process` the state-table $\mathbf{X}'$ to obtain a new state-information structure S.
  $t \leftarrow t + \tau$.
end while

are complete, the marginal distributions are estimated by computing the mean and variance for each species. The iSSA-MPDE follows the system's envelope as it evolves from increment to increment, providing an indication of the system's stochastic stability. If the standard deviation remains small relative to the mean, then the envelope may be regarded as a robust indicator of typical behavior.

`struct` S:	contains a mean vector $S.\boldsymbol{\mu}$ and a standard-deviation vector $S.\boldsymbol{\sigma}$.
`initialize`:	$S.\boldsymbol{\mu} \leftarrow \mathbf{x}_0$ for a given initial state $\mathbf{x}_0$, and $S.\boldsymbol{\sigma} \leftarrow \mathbf{0}$.
`select`:	for each species s_j, generate a noise value n_j from the distribution $\mathcal{N}\left(0,\, S.\sigma_j^2\right)$, and set $x_j \leftarrow S.\mu_j + n_j$.
`record`:	store the k^{th} SSA ending state as $\mathbf{x}'_k$, for $k = 1, ..., K$.
`process`:	compute the sample means and sample variances over all $\mathbf{x}'_k$, and store the results in $S.\boldsymbol{\mu}$ and $S.\boldsymbol{\sigma}$, respectively.

Table 1. Function definitions for iSSA-MPDE.

iSSA-MPDE is derived from the CLE method discussed in Sec. 2, and inherits the τ-leap and CLE conditions[1]. To derive iSSA-MPDE, consider applying the CLE method over a short time-increment τ, beginning at time t with a fixed initial state $\mathbf{x}_0$. At time $t + \tau$, the CLE method returns a state $\mathbf{x}' = \mathbf{x}_0 + \sum_{j=1}^{M} \boldsymbol{\nu}_j$, where each $\boldsymbol{\nu}_j$ is a vector of Gaussian-distributed random values. Because the sum of Gaussians is also Gaussian, the ending state $\mathbf{x}'$ must have a joint Gaussian distribution. Then the distribution of $\mathbf{x}'$ is fully characterized by its mean $\boldsymbol{\mu}$ and its covariance matrix $\boldsymbol{\Gamma}$.

Jointly Gaussian distributions are well understood, and the reaction system's time-evolution can be simulated as the evolution of $\boldsymbol{\mu}$ and $\boldsymbol{\Gamma}$ using the iSSA function definitions shown in Table 2. We refer to this algorithm as *Gaussian probability density evolution* or iSSA-GPDE. A further simplification is possible if the system is represented as a *linear Gaussian network* (LGN), with the form

$$\mathbf{x}' \approx \mathbf{A}\mathbf{x} + \mathbf{n}, \tag{5}$$

where $\mathbf{A}$ is a linear state-transformation matrix and $\mathbf{n}$ is a vector of zero-mean correlated noise with distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma})$. This representation is very close to the linear increment approximation used in general-purpose ODE simulators, including SPICE. The linear Gaussian model provides an intuitively convenient "signal plus noise" representation that is familiar to designers in many disciplines, and may be useful for the design and analysis of biochemical systems.

The computational complexity of this method can be significantly reduced by computing only the marginal statistics, rather than the complete covariance matrix. To compute the marginal statistics, only the diagonal entries of the covariance matrix are computed. Ignoring the remaining terms in $\boldsymbol{\Gamma}$ neglects the statistical dependencies among species in the system. To see when this is allowed, let us examine the system's dependency structure using a Bayesian network model, as shown in Fig. 3. The Bayesian network model contains a column of nodes for each time-index. Within each column, there is a node for each species. Two nodes are

[1] It is possible to apply iSSA-MPDE under a less restrictive set of conditions, but doing so requires a collection of refinements to the method that are beyond the scope of this chapter.

`struct` S:	contains a mean vector $S.\boldsymbol{\mu}$ and a covariance matrix $S.\boldsymbol{\Gamma}$.
`initialize`:	$S.\boldsymbol{\mu} \leftarrow \mathbf{x}_0$ for a given initial state $\mathbf{x}_0$, and $S.\boldsymbol{\Gamma} \leftarrow \mathbf{0}$.
`select`:	generate a correlated noise vector $\mathbf{n}$ from the distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma})$, and set $\mathbf{x} \leftarrow S.\boldsymbol{\mu} + \mathbf{n}$.
`record`:	store the k^{th} SSA ending state as $\mathbf{x}'_k$, for $k = 1, ..., K$.
`process`:	compute the sample mean and sample covariance matrix over all $\mathbf{x}'_k$, and store the results in $S.\boldsymbol{\mu}$ and $S.\boldsymbol{\Gamma}$, respectively.

Table 2. Function definitions for iSSA-GPDE.

connected by an edge if there is a statistical dependency between them. The structure of the Bayesian network is determined by the system's *information matrix*, $\mathbf{J} = \boldsymbol{\Gamma}^{-1}$. An edge (and hence a dependency) exists between nodes x'_a and x'_b if and only if the corresponding entry j_{ab} in J is non-zero (Koller & Friedman, 2009). If $\mathbf{J}$ is approximately diagonal (i.e. if all non-diagonal entries are small relative to the diagonal ones), then the network model contains no edges between any pair $\mathbf{x}'_a, \mathbf{x}'_b$. This means that the *marginal* statistics of $\mathbf{x}'$ are fully determined by the statistics of $\mathbf{x}$. This allows for the joint Gaussian probability distribution at time $t + \tau$ to be approximated as a product of marginal Gaussian distributions. Instead of computing the complete covariance matrix $\boldsymbol{\Gamma}$, it is sufficient to compute the diagonal vector $\boldsymbol{\sigma}$. By computing only marginal statistics in iSSA-GPDE, iSSA-MPDE is obtained, with function definitions shown in Table 1.

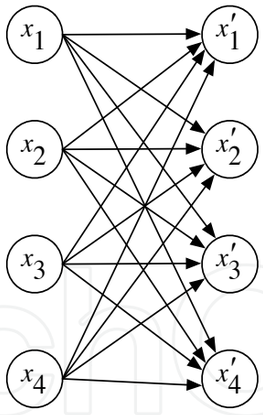


Fig. 3. A linear Gaussian Bayesian network model for a reaction system with four species. Edges in the graph indicate statistical dependencies.

3.3 Conditions and Limitations of iSSA-MPDE

iSSA-MPDE can be interpreted as an instance of belief propagation, with the SSA serving as a Monte Carlo estimate of the species' conditional distributions. When the iSSA-MPDE network is continued over several increments, the corresponding network model is extended, as shown in Fig. 4. When the network is extended in time, loops appear. Some example

loops are indicated by bold edges in Fig. 4. Strictly speaking, belief propagation (and hence iSSA-MPDE) is exact when applied to loop-free Bayesian networks.

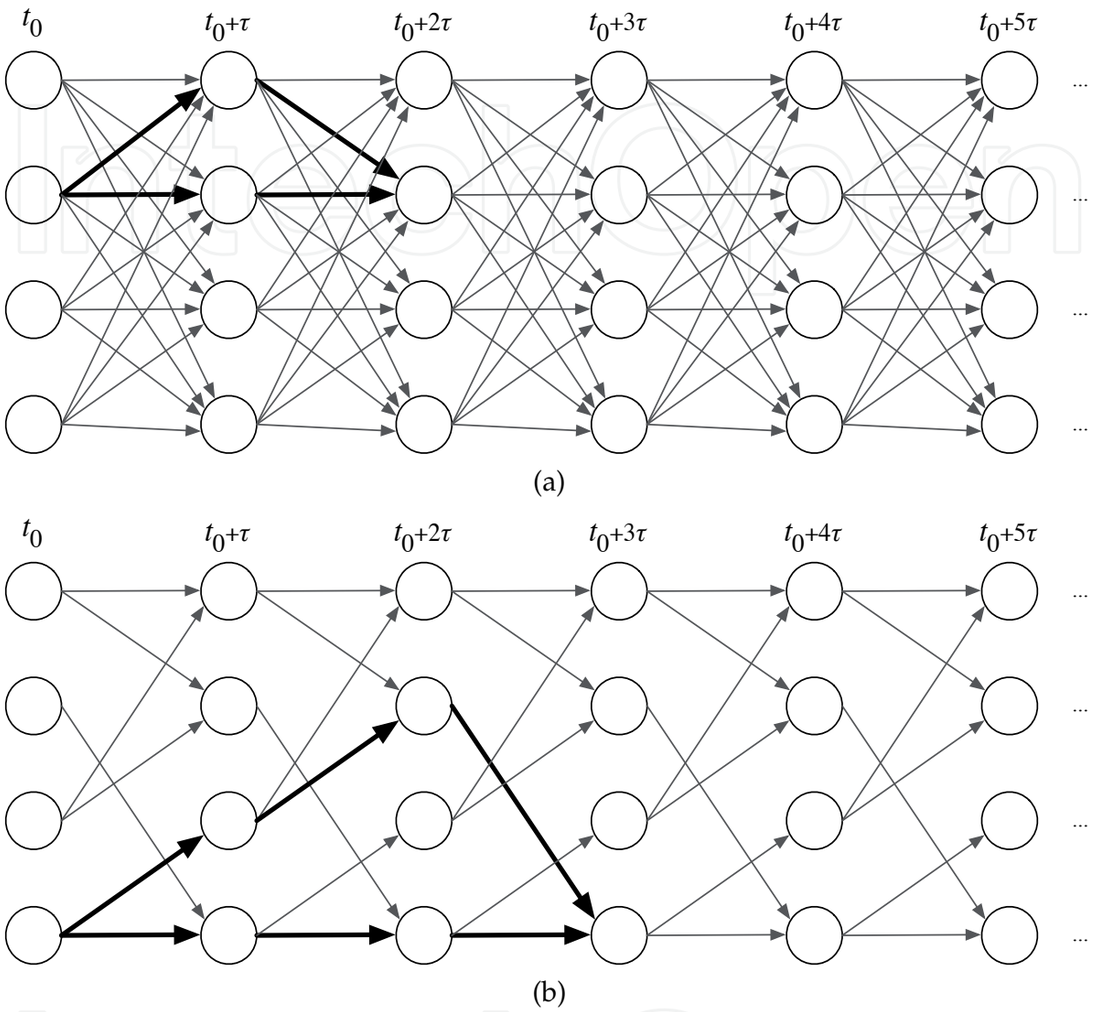


Fig. 4. Loops form when the model is unwrapped across time. (a) A dense reaction model has many short loops. (b) A sparse reaction model has fewer loops, and a larger minimum loop girth.

Loops are unavoidable in reaction network models. As a consequence, iSSA-MPDE corresponds to *loopy* belief propagation, which yields inexact statistical results. Although loopy belief propagation is inexact, it has been shown to provide a close approximation in many application areas (Murphy et al., 1999). The method's accuracy depends on the number of short loops that appear in the graph. An example of a loopy graph is shown in Fig. 4(a). In this graph, there are many loops that allow statistical information to propagate back on top of itself, which distorts the information. A better case is shown in Fig. 4(b), in which there are fewer loops. The highlighted loop in Fig. 4(b) contains six edges. This number is referred to as the loop's girth.

As a general rule, the exactness of loopy belief propagation improves when the minimum loop girth is large. iSSA-MPDE is consequently expected to yield more accurate results for systems with sparse dependencies, as in Fig. 4(b). In networks with dense dependencies, as in Fig. 4(a), iSSA-MPDE may yield distorted results. Large networks of simple reactions (where each reaction contains a small number of reactants and products) tend to be sparse in their dependencies. There are a growing number of *abstraction* methods that reduce the number of effective reactions in a large system and improve the efficiency of simulation. When a system is abstracted in this way, the density of dependencies is unavoidably increased. iSSA-MPDE, therefore, tends to be less attractive for use with abstracted simulation models (Kuwahara et al., 2010; 2006).

3.4 Resolving Variable Dependencies in iSSA-MPDE

In its most basic form, as presented in Table 1, iSSA-MPDE cannot be applied to many important types of reaction systems. This is because many systems have tightly-correlated species which prevent the information matrix from being diagonal. Strong correlations typically arise from conservation constraints, in which the state of one species is completely determined by other states in the system. This section presents a method to identify conservation constraints and correct for their effects in iSSA-MPDE. By resolving conservation constraints, the limitations on iSSA-MPDE can be relaxed considerably, allowing the method to be applied in a broader array of reaction systems.

The circadian rhythm model provides an immediate example of a system with conservation constraints. In this model, the signal molecule A is produced from gene a via transcription/translation reactions. The activity of gene a may be altered by the presence of a repressor molecule R. Hence gene a may be associated with two chemical species, a and a_R, which represent the gene's active and repressed states, respectively. The two states may be represented as distinct species governed by two reactions:

$$a + R \rightarrow a_R, \tag{6}$$

$$a_R \rightarrow a + R \tag{7}$$

In the first of these reactions, the activated gene a is consumed to produce the repressed gene a_R. In the second reaction, the repressed gene is consumed to produce the activated state. At any given time, the gene is in exactly one state. This induces a conservation constraint expressed by the equation $a + a_R = 1$. Since iSSA-MPDE treats a and a_R as independent species, it likely produces states that violate this constraint.

The conservation problem can be resolved if the method is made aware of conservation constraints. Once the constraints are determined, the system may be partitioned into *independent* and *dependent* species. iSSA-MPDE is then executed only on the independent species. The dependent species are determined from the independent ones. This partitioning can be computed automatically at run-time by evaluating the system's stoichiometric matrix, as explained below.

The stoichiometric matrix embodies the network topology of any biochemical system. Several researchers have developed methods for extracting conservation constraints from the stoichiometric matrix (Reder, 1988; Sauro & Ingalls, 2004; Schuster et al., 2002). This section briefly summarizes these techniques and applies them to iSSA-MPDE.

The stoichiometric matrix $\mathbf{N}$ is defined as follows. If a given reaction network is composed of N species and M reactions, then its stoichiometric matrix is an $M \times N$ matrix in which element

a_{ij} equals the net change in species j due to reaction i. In other words, the columns of $\mathbf{N}$ are the state-change vectors $\boldsymbol{\nu}_j$, as defined in Sec. 2.

$$\mathbf{N} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{pmatrix}$$

Conserved cycles in a chemical reaction network appear as linear dependencies in the row dimensions of the stoichiometric matrix. In systems where conservation constraints appear, the sum of the conserved species must be constant. For example, consider a conservation law of the form $s_1 + s_2 = k$ for some constant k. This law dictates that the rate of appearance of s_1 must equal the rate of disappearance of s_2. Mathematically, this condition is expressed as

$$\frac{dS_1}{dt} + \frac{dS_2}{dt} = 0 \tag{8}$$

When conservation relationships are present in a biochemical network, there are linearly dependent rows in the stoichiometric matrix. Following the notation in Sauro & Ingalls (2004), one can partition the rows of $\mathbf{N}$ into two sections, $\mathbf{N_R}$ and $\mathbf{N_0}$, which represent independent and dependent species, respectively. Thus, one can partition $\mathbf{N}$ as follows:

$$\mathbf{N} = \begin{bmatrix} \mathbf{N_R} \\ \mathbf{N_0} \end{bmatrix} \tag{9}$$

Since $\mathbf{N_0}$ is a function of $\mathbf{N_R}$, the concentrations of the independent species, $\mathbf{N_R}$, can be used to calculate those of the dependent species $\mathbf{N_0}$. This relationship is determined by the *link-zero* matrix, defined as the matrix $\mathbf{L_0}$ which satisfies

$$\mathbf{N_0} = \mathbf{L_0} \times \mathbf{N_R} \tag{10}$$

Equations (9) and (10) can be combined to yield

$$\mathbf{N} = \begin{bmatrix} \mathbf{N_R} \\ \mathbf{L_0 N_R} \end{bmatrix} \tag{11}$$

Equation (11) can be further reduced by combining $\mathbf{L_0}$ with an identity matrix $\mathbf{I}$ and taking $\mathbf{N_R}$ as a common factor outside of the brackets, as shown in Equation (12).

$$\mathbf{N} = \begin{bmatrix} \mathbf{I} \\ \mathbf{L_0} \end{bmatrix} \mathbf{N_R} \tag{12}$$

$$\mathbf{N} = \mathbf{L N_R}, \tag{13}$$

where $\mathbf{L} = [\mathbf{I} \quad \mathbf{L_0}]^{\mathbf{T}}$ is called the *link* matrix. For systems in which conservation relationships do not exist, $\mathbf{N} = \mathbf{N_R}$, thus $\mathbf{L} = \mathbf{I}$.

Based on this analysis, the species are partitioned into independent and dependent state vectors, $\mathbf{s}_i(t)$ and $\mathbf{s}_d(t)$, respectively. Due to the conservation laws, any change in $\mathbf{s}_i$ must be compensated by a corresponding change in $\mathbf{s}_d$, hence

$$\mathbf{s}_d(t) - \mathbf{L_0 s}_i(t) = \mathbf{s}_d(0) - \mathbf{L_0 s}_i(0)), \tag{14}$$

If the initial condition is given and the link-zero matrix is known, then the dependent species can always be computed from the independent species. To compute the link-zero matrix, we observe that

$$\begin{bmatrix} -\mathbf{L_0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{N_R} \\ \mathbf{N_0} \end{bmatrix} = \mathbf{0}. \tag{15}$$

This equation reveals that $[-\mathbf{L_0}\mathbf{I}]$ is the left null-space of $\mathbf{N}$. There are a variety of ways to compute the null-space of a matrix, and most numerical tools have built-in functions for this purpose.

iSSA-MPDE can be applied to systems with conservation constraints if the system is suitably partitioned into independent and dependent species. The partitioning is done automatically by identifying the linearly independent rows of the stoichiometric matrix $\mathbf{N}$, which correspond to the independent species in the system. The link-zero matrix is then computed as part of the simulation's initialization. During execution of the iSSA algorithm, the MPDE method is applied only to the independent species. The dependent species are generated using (14). Using this approach, the independent species must satisfy the conditions and limitations discussed above. The dependent species only need to satisfy the conservation constraints expressed by (14).

To demonstrate the MPDE method with constraint resolution, the method was applied to the circadian rhythm model. The results are shown in Fig. 5. The results obtained using this method agree well with the pattern observed in SSA simulations. The MPDE results also reveal the typical characteristics of the circadian rhythm system, which are difficult to discern from the SSA simulation results shown in Fig. 2.

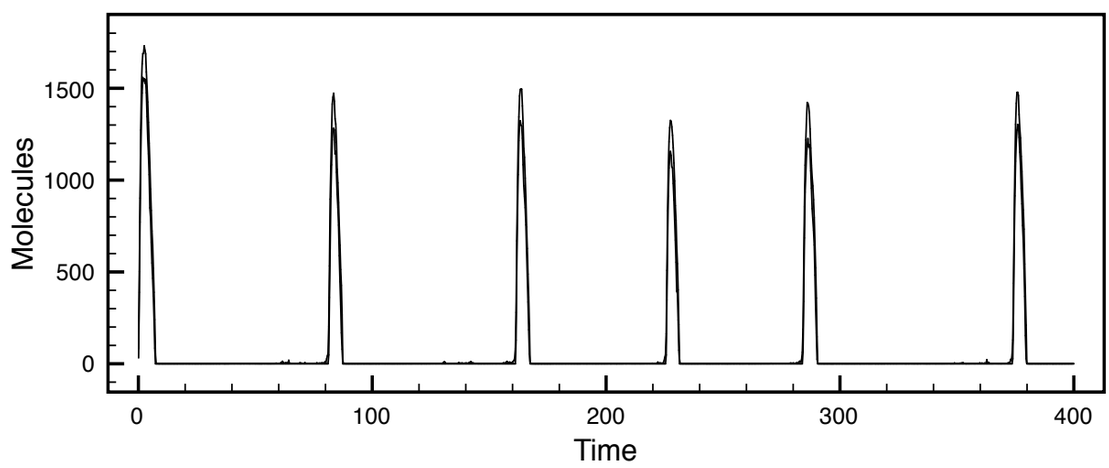


Fig. 5. The circadian rhythm model simulated using iSSA-MPDE with constraint resolution.

4. Rare Deviant Event Analysis

While the previous section discusses how to determine typical behavior, this section describes a method for more efficiently determine the likelihood of rare events. In robust biological systems, wide deviations from highly controlled normal behavior may occur with extremely small probability; nevertheless, they can have significant influences and profound consequences in many systems (Csete & Doyle, 2004). This is particularly true in biochemical and

`struct` S:	contains a mean vector $S.\boldsymbol{\mu}$ and a standard-deviation vector $S.\boldsymbol{\sigma}$.
`initialize`:	$S.\boldsymbol{\mu} \leftarrow \mathbf{x}_0$ for a given initial state $\mathbf{x}_0$, and $S.\boldsymbol{\sigma} \leftarrow \mathbf{0}$. Independent species are identified from the stoichiometric matrix $\mathbf{N}$. The link-zero matrix $\mathbf{L_0}$ is computed using (15).
`select`:	for each *independent* species s_j, generate a noise value n_j from the distribution $\mathcal{N}\left(0,\, S.\sigma_j^2\right)$, and set $x_j \leftarrow S.\mu_j + n_j$. Compute the remaining dependent species using the conservation law (14).
`record`:	store the k^{th} SSA ending state as $\mathbf{x}'_k$, for $k = 1, ..., K$.
`process`:	compute the sample means and sample variances for each of the independent species in $\mathbf{x}'$, and store the results in $S.\boldsymbol{\mu}$ and $S.\boldsymbol{\sigma}$, respectively.

Table 3. Function definitions for the MPDE-iSSA method with resolved conservation constraints.

physiological systems in that, while the occurrence of biochemical events that leads to some abnormal states may be rare, it can have devastating effects. In order to study the underlying mechanisms of such rare yet catastrophic events *in silico*, computational simulation methods may become a useful tool. However, computational analysis of rare events can demand significant computational costs and, even for a relatively small SCK model, computational requirements for a rare event analysis with the SSA may exceed the power of the most current computers. This section presents a simulation method for rare event analysis called *weighted SSA* (wSSA) (Kuwahara & Mura, 2008). Section 4.1 first defines the properties of interest and their computational challenges. Section 4.2 then briefly discusses the theoretical basis of the wSSA. Finally, Section 4.3 presents the algorithm in detail.

4.1 Background

Traditionally, analysis of rare events has been associated with analysis of the first passage time distribution (Gillespie et al., 2009), and considerable attention has been directed towards making the analysis of the first passage time to reach a rare event of interest more efficient (e.g., Allen et al. (2006); Misra & Schwartz (2008)). This section formulates rare event analysis rather differently from the analysis of the first passage time in that the property of interest here is the time-bounded probability of $\mathbf{X}(t)$ reaching a certain subset of states given that the process $\mathbf{X}(t)$ starts from a different state. In other words, our objective is to analyze $P_{t \leq t_{max}}(\mathbf{X} \rightarrow \mathcal{E} \mid \mathbf{x_0})$, the probability that $\mathbf{X}$ moves to a state in a subset states $\mathcal{E}$ within time limit t_{max}, given $\mathbf{X}(0) = \mathbf{x_0}$ where $\mathbf{x_0} \notin \mathcal{E}$, specifically when $P_{t \leq t_{max}}(\mathbf{X} \rightarrow \mathcal{E} \mid \mathbf{x_0})$ is very small. This type of time-bounded rare event analyses may be very useful when it comes to study of specific biological events of interest per cell generation (i.e., before protein and RNA molecules in a mother cell are partitioned via cell division).

A standard way to analyze $P_{t \leq t_{max}}(\mathbf{X} \rightarrow \mathcal{E} \mid \mathbf{x_0})$ is to define a Boolean random variable Y such that $Y = 1$ if $\mathbf{X}(t)$ moves to some states in $\mathcal{E}$ within the time limit and $Y = 0$ otherwise. Then, the average of Y gives $P_{t \leq t_{max}}(\mathbf{X} \rightarrow \mathcal{E} \mid \mathbf{x_0})$. Thus, with the SSA, $P_{t \leq t_{max}}(\mathbf{X} \rightarrow \mathcal{E} \mid \mathbf{x_0})$ can be estimated by generating n samples of Y: $Y_1, \ldots, Y_n$ through n simulation runs of $\mathbf{X}(t)$, and taking the sample average: $1/n \sum_{i=1}^{n} Y_i$. Chief among the problems in this statistical approach to project the probability of a rare event is that it may require a large number of simulation

runs just to observe the first few instances of the rare event of interest. For example, the spontaneous, epigenetic switching rate from the lysogenic state to the lytic state in phage λ-infected *Escherichia coli* (Ptashne, 1992) is experimentally estimated to be in the order of 10^{-7} per cell per generation (Little et al., 1999). Thus, simulation of one cell generation via the SSA would expect to generate sample trajectories of this rare event only once every 10^7 runs, and it would require more than 10^{11} simulation runs to generate an estimated probability with a 95 percent confidence interval with 1 percent relative half-width. This indicates that the computational requirements for obtaining results at a reasonable degree of statistical confidence can be substantial as the number of samples needed for such results may be astronomically high. Furthermore, this highlights the fact that computational requirements involved in rare event analysis of even a relatively simple biological system can far exceed the ability of most computers.

4.2 Theoretical Basis of the wSSA

The wSSA (Kuwahara & Mura, 2008) increases the chance of observing the rare events of interest by utilizing the *importance sampling* technique. Importance sampling manipulates the probability distribution of the sampling so as to observe the events of interest more frequently than it would otherwise with the conventional Monte Carlo sampling. The outcome of each biased sampling is weighted by a likelihood factor to yield the statistically correct and unbiased results. Thus, the importance sampling approach can increase the fraction of samples that result in the events of interest per a given set of simulation runs, and consequently, it can efficiently increase the precision of the estimated probability. An illustrative example of importance sampling is depicted in Figure 6.

By applying importance sampling to simulation of SCK models, hence, the wSSA can substantially increase the frequency of observation of the rare events of interest, allowing reasonable results to be obtained with orders of magnitude smaller simulation runs than the SSA. This can result in a substantial increase in computational efficiency of rare event analysis of biochemical systems.

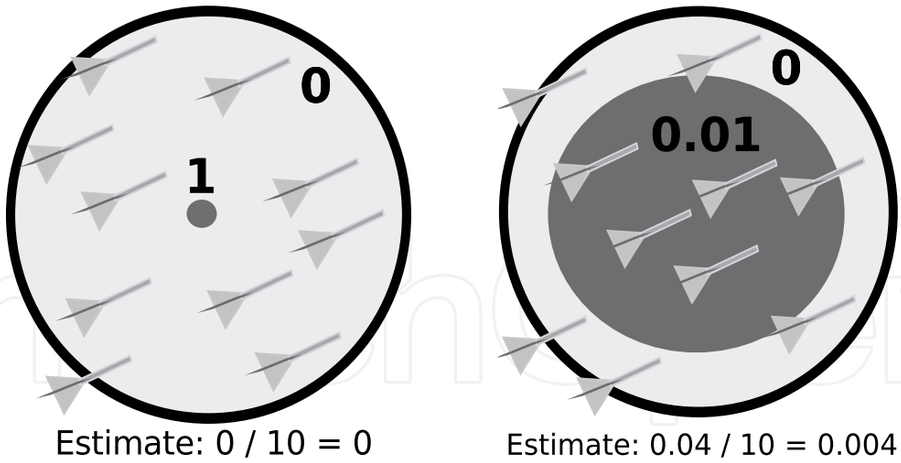
In order to observe reaction events that can lead to a rare event of interest more often, for each reaction R_j, the wSSA utilizes *predilection function* $b_j(\mathbf{x})$ to select the next reaction instead of utilizing the propensity function $a_j(\mathbf{x})$. The predilection functions are defined such that $b_j(\mathbf{x})dt$ is the probability with which, given $\mathbf{X} = \mathbf{x}$, one R_j reaction event should occur within the next infinitesimal time dt, based on the bias one might have to lead $\mathbf{X}(t)$ towards the events of interest. With the definition of predilection functions, the index of the next reaction selection is sampled with the following probability:

$$Prob\{\text{the next reaction index is } j \text{ given } \mathbf{X} = \mathbf{x}\} = \frac{b_j(\mathbf{x})}{b_0(\mathbf{x})},$$

where $b_0(\mathbf{x}) \equiv \sum_{\mu=1}^{M} b_\mu(\mathbf{x})$. To correct the sampling bias in the reaction selection and yield the statistically unbiased results, each weighted reaction selection is then weighted by the weight function:

$$w(j, \mathbf{x}) = \frac{a_j(\mathbf{x})b_0(\mathbf{x})}{a_0(\mathbf{x})b_j(\mathbf{x})}.$$

Now, consider a k-jump trajectory of $\mathbf{X}(t)$, and let $P_k(j_k, k; \cdots; j_2, 2; j_1, 1 \mid \mathbf{x_0})$ denote the probability that, given $\mathbf{X} = \mathbf{x_0}$, the first reaction is R_{j_1}, the second reaction is R_{j_2},..., and the k-th



(a) (b)

Fig. 6. An illustrative example for importance sampling. Here, the probability of hitting the area of the dart board is uniformly distributed, and the objective is to estimate the fraction of the dark grey area, which is 0.005, by throwing ten darts. (a) With the standard approach, each dart scores 1 if it hits the dark grey area and 0 otherwise. In this example, since no hit is observed in ten darts, the estimate becomes 0. (b) With the importance sampling approach, here, the dark grey area is enlarged 100 times to observe more hits and the score of the dark grey area is reduced by 100 times to correct the unbiased results. In this example, since four among the 10 darts hit the dark grey area, the estimate becomes 0.004, which is substantially closer to the true value than the original estimate.

reaction is R_{j_k}. Then, since $\mathbf{X}(t)$ is Markovian, this joint conditional probability can be expressed as follows:

$$P_k(j_k, k; \cdots; j_2, 2; j_1, 1 \mid \mathbf{x_0}) = \prod_{h=1}^{k} \frac{a_{j_h}(\mathbf{x_{h-1}})}{a_0(\mathbf{x_{h-1}})} \tag{16}$$

where $\mathbf{x_h} = \mathbf{x_0} + \sum_{h'=1}^{h-1} \mathbf{v_{j_{h'}}}$. Equation 16 can also be expressed in terms of the weight functions and the predilection functions as follows:

$$\begin{aligned} P_k(j_k, k; \cdots; j_2, 2; j_1, 1 \mid \mathbf{x_0}) &= \prod_{h=1}^{k} \left[\frac{a_{j_h}(\mathbf{x_{h-1}}) b_0(\mathbf{x_{h-1}})}{b_{j_h}(\mathbf{x_{h-1}}) a_0(\mathbf{x_{h-1}})} \right] \frac{b_{j_h}(\mathbf{x_{h-1}})}{b_0(\mathbf{x_{h-1}})} \\ &= \prod_{h=1}^{k} w(j_h, \mathbf{x_{h-1}}) \prod_{h=1}^{k} \frac{b_{j_h}(\mathbf{x_{h-1}})}{b_0(\mathbf{x_{h-1}})}. \end{aligned} \tag{17}$$

Hence, in the wSSA, the estimate of $P_{t \le t_{max}}(\mathbf{X} \to \mathcal{E} \mid \mathbf{x_0})$ is calculated by first defining the statistical weight of the i-th sample trajectory w_i such that

$$w_i = \begin{cases} \prod_{h=1}^{k_i} w(j_h, \mathbf{x_{h-1}}) & \text{if } \mathbf{X}(t) \text{ moves to some state in } \mathcal{E} \text{ within the time limit,} \\ 0 & \text{otherwise,} \end{cases}$$

where k_i is the number of jumps in the i-th sample trajectory. Then, $P_{t \leq t_{max}}(\mathbf{X} \rightarrow \mathcal{E} \mid \mathbf{x_0})$ is estimated by taking a sample average of w_i:

$$\frac{1}{n} \sum_{i=1}^{n} w_i.$$

With an adequate choice of the predilection functions, the wSSA can increase the fraction of sample trajectories that result in the rare events of interest. At the same time, it can lower the variance of the estimate by having each w_i smaller than 1.
In Kuwahara & Mura (2008), each predilection function has a restricted form in that each predilection function is proportional to the corresponding propensity function. In other words, for each reaction R_j, $b_j(\mathbf{x})$ is defined as:

$$b_j(\mathbf{x}) = \alpha_j \times a_j(\mathbf{x}), \tag{18}$$

where each $\alpha_j > 0$ is a constant. This restriction can conveniently constrain the predilection functions such that, for each $b_j(\mathbf{x})$, $b_j(\mathbf{x}) = 0$ if and only if $a_j(\mathbf{x}) = 0$, avoiding the case where a possible trajectory of a system is weighted by a factor 0. Clearly, if $\alpha_j = \alpha$ for all j, then $a_j(\mathbf{x})/a_0(\mathbf{x}) = b_j(\mathbf{x})/b_0(\mathbf{x})$. Thus, such a selection of predilection functions may not be useful. Nevertheless, the wSSA can substantially accelerate the analysis of rare events when appropriate predilection functions are used. While optimized selection schemes of the predilection functions require further investigation, it is somewhat intuitive to select predilection functions to alleviate the computational demands in a number of cases. For example, suppose we are interested in analyzing the probability that a species S transitions from θ_1 to θ_2 where $\theta_1 < \theta_2$. Then, most likely, increasing the predilection functions of the production reactions of S and/or decreasing the predilection functions of the degradation reactions of S—even with a small factor—would increase the fraction of the sample trajectories that result in the event of interest. Furthermore, a procedure to choose optimized α_j by running several test runs to compute the variance of the statistical weights has been proposed (Gillespie et al., 2009). However, much work remains to be done in order to more practically select predilection functions.

4.3 Algorithm of the wSSA

Algorithm 5 describes the procedure to estimate $P_{t \leq t_{max}}(\mathbf{X} \rightarrow \mathcal{E} \mid \mathbf{x_0})$ with n simulation runs of the wSSA. Note that, while Algorithm 5 is presented in a similar fashion as the counterpart direct method of the SSA, various optimization techniques of the direct method, such as Cao et al. (2004); McCollum et al. (2006), can also be applied to an implementation of the wSSA to further reduce the simulation cost. Furthermore, model abstraction techniques such as Kuwahara & Myers (2007) can be incorporated to further accelerate the simulation process.
First, the algorithm initializes to 0 the variable q, which accumulates statistical weights of each successful sample trajectory (line 1). Then, it generates n sample trajectories of $\mathbf{X}(t)$ via the wSSA. For each simulation run, the initialization is first performed to set the weight of each sample trajectory, w, the time, t, and the system state, $\mathbf{x}$ to 1, 0, and $\mathbf{x_0}$, respectively (line 3). It then evaluates all the propensity functions $a_j(\mathbf{x})$ and all the predilection functions $b_j(\mathbf{x})$, and also calculates $a_0(\mathbf{x})$ and $b_0(\mathbf{x})$ (line 4). Each Monte Carlo simulation is run up to time t_{max}. If, however, a rare event (i.e., $\mathbf{x} \in \mathcal{E}$) occurs within t_{max}, then the current sample trajectory weight w is added to q, and the next simulation run is performed (lines 6-9). Otherwise, the waiting time to the next reaction, τ, is sampled in the same way as in the direct method of the SSA, while the next reaction R_μ is selected using the predilection functions (lines 10-12). Then,

w, t, and $\mathbf{x}$ are updated to reflect the selections of the waiting time and the next reaction (lines 13-15). Any propensity functions and predilection functions that need to be updated based on the firing of one R_μ reaction event are re-evaluated, and $a_0(\mathbf{x})$ and $b_0(\mathbf{x})$ are re-calculated (line 16). After n sample trajectories are generated via the wSSA, the probability that $\mathbf{X}(t)$ reaches some state in $\mathcal{E}$ within t_{max} given $\mathbf{X}(0) = \mathbf{x_0}$ is estimated by q/n (line 19).

Algorithm 5 Estimate of $P_{t \le t_{max}}(\mathbf{X} \to \mathcal{E} \mid \mathbf{x_0})$ via wSSA

1: $q \leftarrow 0$
2: **for** $k = 1$ to n **do**
3: $w \leftarrow 1, t \leftarrow 0, \mathbf{x} \leftarrow \mathbf{x_0}$
4: evaluate all $a_j(\mathbf{x})$ and $b_j(\mathbf{x})$, and calculate $a_0(\mathbf{x})$ and $b_0(\mathbf{x})$
5: **while** $t \le t_{max}$ **do**
6: **if** $\mathbf{x} \in \mathcal{E}$ **then**
7: $q = q + w$
8: break out of the while loop
9: **end if**
10: $\tau \leftarrow$ a sample of exponential random variable with mean $1/a_0(\mathbf{x})$
11: $u \leftarrow$ a sample of unit uniform random variable
12: $\mu \leftarrow$ smallest integer satisfying $\sum_{i=1}^{\mu} b_i(\mathbf{x}) \ge u b_0(\mathbf{x})$
13: $w \leftarrow w \times \left(a_\mu(\mathbf{x})/b_\mu(\mathbf{x})\right) \times (b_0(\mathbf{x})/a_0(\mathbf{x}))$
14: $t \leftarrow t + \tau$
15: $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}$-
16: update $a_j(\mathbf{x})$ and $b_j(\mathbf{x})$, and re-calculate $a_0(\mathbf{x})$ and $b_0(\mathbf{x})$
17: **end while**
18: **end for**
19: report q/n as the estimated probability

The computational complexity of Algorithm 5 and the counterpart of the standard SSA can be compared by noticing that the multiplication/division operations in the wSSA only increases linearly. Indeed, the operation count in Algorithm 5 differs from the counterpart of the SSA only in the two steps: line 13; and line 16 inside the **while** loop. Line 13 adds a constant number of operations (i.e., 2 multiplications and 2 divisions), while line 16 includes the operations for the update of the predilection functions $b_j(\mathbf{x})$, $j = 1, 2, \ldots, M$ as well as $b_0(\mathbf{x})$. The cost of such updates depends on the specific form of the predilection functions and the network of the model. However, if, as considered in this section, the predilection functions take the form of simple scaling functions of the propensity functions, then these updates require at most M multiplications, which does not change the overall complexity of the presented simulation algorithm between the wSSA and the direct method of the SSA.

5. Case Study: Enzymatic Futile Cycles

This section presents case studies of the two simulation methods described in this chapter to illustrate the usefulness of those methods. Our case studies are based on the analysis of dynamical properties of enzymatic futile cycle models. Section 5.1 introduces the structure of an enzymatic futile cycle model. Section 5.2 shows iSSA results on the futile cycle model. Finally, Section 5.3 shows the results from wSSA-based rare event analysis on this model.

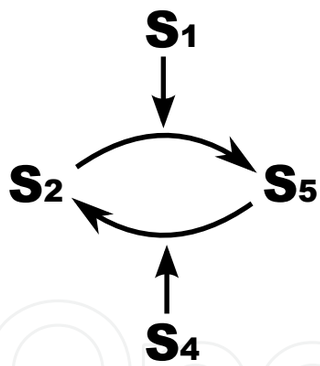
5.1 Enzymatic Futile Cycle Model

The enzymatic futile cycle is composed of two enzymatic reactions running opposite directions, and is ubiquitously seen in biological systems (Voet et al., 1999). In signaling networks, for example, this control motif can be used as a biological network building block that regulates the activity of a protein by representing a phosphorylation-dephosphorylation cycle where the forward enzymatic reaction represents the phosphorylation of a protein via a kinase or an activation of a protein via a small GTP-binding protein, while the backward enzymatic reaction represents the dephosphorylation of the protein via phosphatase (Goldbeter & Koshland, 1981). A three-layered cascade of phosphorylation-dephosphorylation cycles can form the basic structure of the mitogen-activated protein kinase cascade, which facilitates generation of a variety of responses to external stimuli and is ubiquitously seen in eukaryotes to control many biological processes including cell proliferation and apoptosis (Chang & Karin, 2001; Huang & Ferrell, 1996).

The structure of an enzymatic futile cycle model is depicted in Figure 7. This model has six species: S_1 is the enzyme to catalyze the transformation of the protein into the active form; S_2 is the inactive form of the protein; S_3 is the complex of S_1 and S_2; S_4 is the enzyme to catalyze the transformation of the protein into the inactive form; S_5 is the active form of the protein; and S_6 is the complex of S_4 and S_5 (Figure 7(a)). The model has six reactions: R_1 is the formation of S_3; R_2 is the breakup of S_3 into S_1 and S_2; R_3 is the production of S_5; R_4 is the formation of S_6; R_4 is the breakup of S_6 into S_4 and S_5; and R_6 is the production of S_2. A schematic of this model is shown in Figure 7(b).

$$
\begin{aligned}
R_1 &: \quad \mathrm{S}_1 + \mathrm{S}_2 \xrightarrow{c_1} \mathrm{S}_3 \\
R_2 &: \quad \mathrm{S}_3 \xrightarrow{c_2} \mathrm{S}_1 + \mathrm{S}_2 \\
R_3 &: \quad \mathrm{S}_3 \xrightarrow{c_3} \mathrm{S}_1 + \mathrm{S}_5 \\
R_4 &: \quad \mathrm{S}_4 + \mathrm{S}_5 \xrightarrow{c_4} \mathrm{S}_6 \\
R_5 &: \quad \mathrm{S}_6 \xrightarrow{c_5} \mathrm{S}_4 + \mathrm{S}_5 \\
R_6 &: \quad \mathrm{S}_6 \xrightarrow{c_6} \mathrm{S}_4 + \mathrm{S}_2
\end{aligned}
$$

(a)



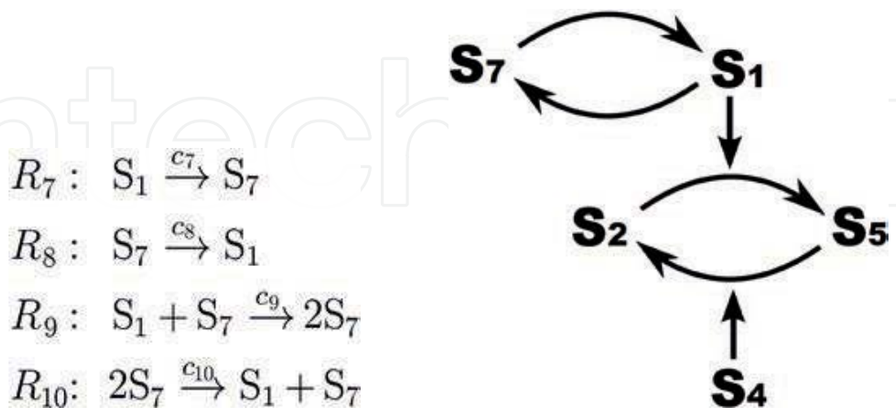
(b)

Fig. 7. The structure of an enzymatic futile cycle model. Here, S_1 is the enzyme to catalyze the transformation of S_2 into S_5, while S_4 is the enzyme to catalyze the transformation of S_5 into S_2. S_3 is the complex of S_1 and S_2. S_6 is the complex of S_4 and S_5 (a) A list of the six reactions in the model. (b) A schematic of the enzymatic futile cycle model.

5.2 Bistable Oscillation in Enzymatic Futile Cycles with Noise Driver

To demonstrate the utility of the iSSA, this section considers an enzymatic futile cycle model with a noise driver as shown in Fig. 8 (Samoilov et al., 2005). This model has the same two enzymatic reactions as the original futile cycle model but also includes a species S_7 and four

more reactions that involve S_1 and S_7 in order to simulate noise in the environment. R_7 converts S_1 into S_7; R_8 is the reverse reaction of R_7 and converts S_7 back into S_1; R_9 converts S_1 and S_7 into two S_7; and R_{10} is the reverse reaction of R_9 and converts two S_7 back into S_1 and S_7.



(a) (b)

Fig. 8. Model for enzymatic futile cycle with a noise driver. Here the species S_7 has been added to introduce noise on the amount of S_1 available. This model also includes four additional reactions that convert between S_1 and S_7 molecules.

Simulation results for this model are expected to result in random symmetric oscillations of species S_2 and S_5 as depicted in the individual SSA run shown in Figure 9(a). However, Figure 9(b) shows that when 10 SSA runs are averaged together, S_2 and S_5 clearly do not exhibit this behavior and potentially leading to the conclusion that this model does not oscillate. When iSSA is applied to this model, the results reveal the expected oscillatory behavior as shown in Figures 9(c). These plots present the results for 10 runs for each time increment and a time step of 0.01. These results show that drawing conclusions from aggregate SSA statistics is problematic. The iSSA, on the other hand, aggregates stochastic run statistics in small time increments in order to produce typical behavior profiles of genetic circuits.

5.3 Rare Event Analysis in Balanced Enzymatic Futile Cycles

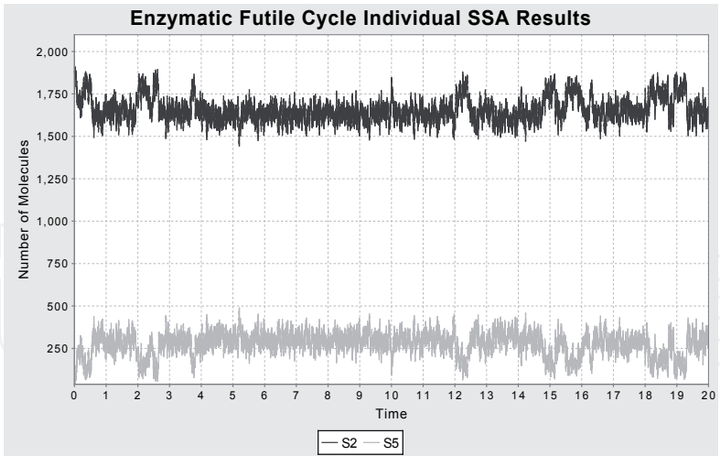
To illustrate the utility of wSSA, this section considers a balanced enzymatic futile cycle model and aims at evaluating $P_{t\leq 100}(X_5 \to 25 \mid \mathbf{x_0})$, the probability that, given $\mathbf{X}(0) = \mathbf{x_0}$, X_5 moves to 25 within 100 time units. In this study, the initial state of the enzymatic futile cycle model is given by

$$X_1(0) = X_4(0) = 1; X_2(0) = X_5(0) = 50; \text{ and } X_3(0) = X_6(0) = 0,$$

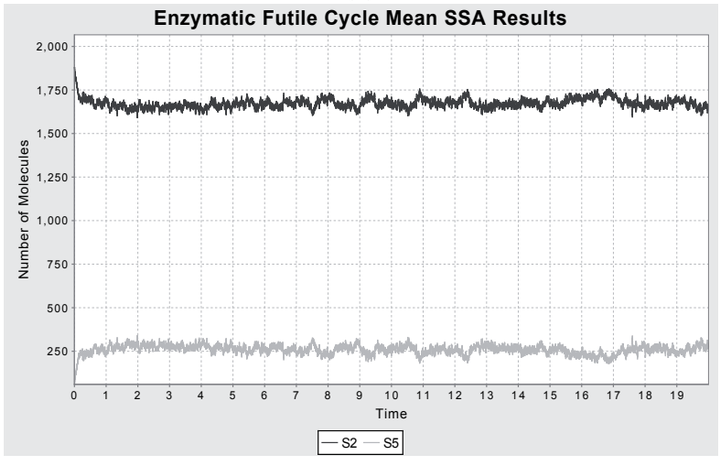
and the rate constants are specified as follows:

$$k_1 = k_2 = k_4 = k_5 = 1; \text{ and } k_3 = k_6 = 0.1.$$

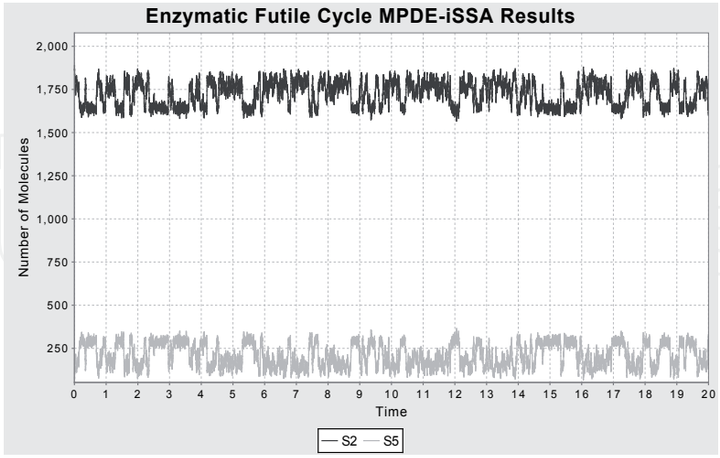
Because of the perfect symmetry in the rate constants as well as in the initial molecule counts of the two enzymatic reactions in this setting, $\mathbf{X}(t)$ tends to stay—with high probability—around states in which X_2 and X_5 are balanced from time 0. That is, X_2 and X_5 stay around 50.



(a)



(b)



(c)

Fig. 9. SSA simulation results for S_2 and S_5 from the enzymatic futile cycle with noise driver. (a) A single SSA sample path. (b) The mean $\overline{x}(t)$ of 10 independent SSA sample paths. (c) iSSA results using 10 runs for each time increment and a time step of 0.01.

This implies that $X_5 \rightarrow 25 \mid \mathbf{x_0}$ is a rare deviant event. As the underlying Markov process has a finite and relatively small number of states, we have computed the exact value of $P_{t\leq 100}(X_5 \rightarrow 25 \mid \mathbf{x_0})$ through a numerical solution, which in turn serves as the measure to compare the accuracy of the wSSA and the SSA.
In order to increase the fraction of simulation runs that reach of the states of interest in the wSSA for this analysis, the following predilection functions are used:

$$b_j(\mathbf{x}) = \begin{cases} a_j(\mathbf{x}) & \text{for } j = 1, 2, 4, 5, \\ \gamma a_j(\mathbf{x}) & \text{for } j = 3, \\ \frac{1}{\gamma} a_j(\mathbf{x}) & \text{for } j = 6, \end{cases}$$

where $\gamma = 0.5$. This biasing approach discourages the forward enzymatic reaction while encourages the backward enzymatic reaction, resulting in an increase in the likelihood of X_5 to move to low count states.
Figure 10 depicts the accuracy of the estimates of $P_{t\leq 100}(X_5 \rightarrow 25 \mid \mathbf{x_0})$ via the SSA and the wSSA with respect to a number of simulation runs. In the SSA, we did not observe any simulation runs that had resulted in X_5 moving to 25 within 100 time units for the first 10^5 simulation runs, making the estimated probability 0 (Figure 10(a)). On the other hand, wSSA was able to produce a reasonable estimate in the first 100 simulation runs and, throughout, it generated an estimated probability which is in very close agreement with the true probability (Figure 10(a)). Furthermore, the relative distance of the estimate from the true value indicates that the estimate from the wSSA can converge to the true value more rapidly than that from the SSA (Figure 10(b)).

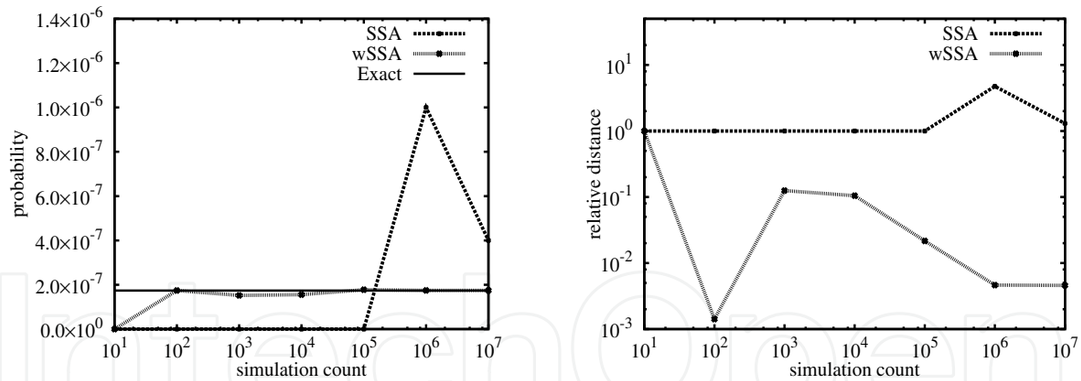
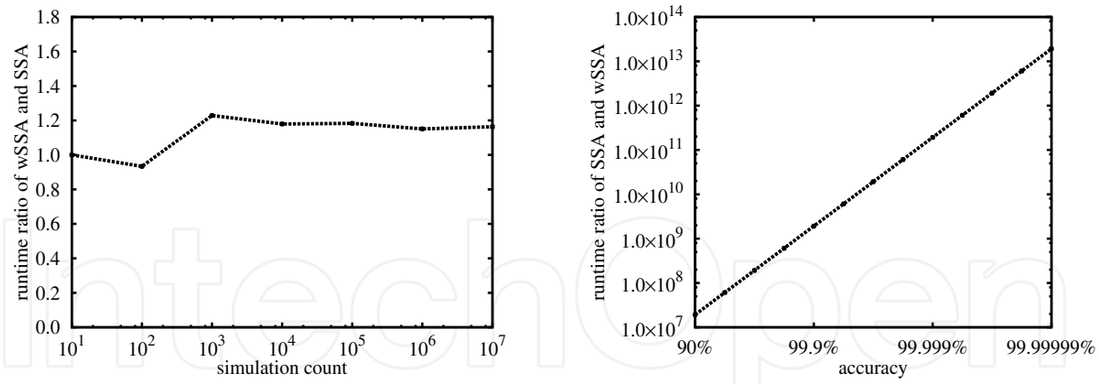


Fig. 10. Comparison of accuracy between SSA and wSSA for the estimate of $P_{t\leq 100}(X_5 \rightarrow 25 \mid \mathbf{x_0})$. (a) The estimated probability via the SSA and the wSSA with respect to a number of simulation runs. The solid line represents the true probability. (b) The relative distance of the estimated probability from the true value with respect to a number of simulation runs.

The ratio of the simulation time between the wSSA and the SSA with respect to a number of simulation runs is illustrated in Figure 11(a). This shows that, in the worst case, the run time of wSSA is about 1.2 times slower than the direct method of the SSA. However, since the wSSA achieved orders of magnitude higher accuracy in estimate of $P_{t\leq 100}(X_5 \rightarrow 25 \mid \mathbf{x_0})$ than



(a) (b)

Fig. 11. Comparison of computation efficiency between SSA and wSSA for the estimate of $P_{t\leq 100}(X_5 \rightarrow 25 \mid \mathbf{x_0})$. The ratio of the simulation time of the wSSA and the SSA with respect to a number of simulation runs. (b) Ratio of SSA and wSSA computation time for a given level of accuracy.

the SSA per a given number of simulation runs, the wSSA is substantially more efficient than the SSA in computing a high precision estimate of $P_{t\leq 100}(X_5 \rightarrow 25 \mid \mathbf{x_0})$.
To better characterize the computational gain obtained with the wSSA over the SSA, we evaluated the number of runs required by SSA to achieve a given accuracy criterion ϵ where ϵ is defined as 1 minus the relative distance of the estimate from the true probability. We then estimated the number of simulation runs required by the SSA through a statistical argument based on confidence intervals (see Appendix of Kuwahara & Mura (2008) for details). By factoring in the estimated number of runs and the average run time, we computed the expected computation time of SSA for given ϵ. Figure 11(b) shows the ratio of the expected computation time between the SSA and wSSA. This illustrates a significant computational gain that is achieved via the wSSA. For instance, while the wSSA can estimate $P_{t\leq 100}(X_5 \rightarrow 25 \mid \mathbf{x_0})$ with an accuracy of 99.9999% in 1.7×10^3 seconds, the SSA would need 10^{12} times as much computational time, which is roughly 1.05×10^8 years of computation (i.e., 2.2×10^{19} simulation runs) to achieve that same level of accuracy on the same computer.

6. Conclusions

During stochastic analysis of biological systems, it is important to be able to both determine accurately and efficiently the typical behavior and the probability of rare deviant events. This chapter has introduced two new stochastic simulation algorithms, the iSSA and wSSA, to address these problems. The iSSA has been shown to produce a more stable typical behavior of an oscillatory system than aggregate statistics generated by the traditional SSA. The wSSA has been shown to produce a substantially more accurate estimate of the probability of rare deviant events as compared to same number of runs of the SSA. Taken together, these are powerful tools for the analysis of biological systems.

7. References

Allen, R. J., Frenkel, D. & ten Wolde, P. R. (2006). Forward flux sampling-type schemes for simulating rare events: Efficiency analysis, *The Journal of Chemical Physics* **124**(19): 194111.
URL: *http://link.aip.org/link/?JCP/124/194111/1*

Arkin, A. & Fletcher, D. (2006). Fast, cheap and somewhat in control, *Genome Biology* **7**(8): 114.
URL: *http://genomebiology.com/2006/7/8/114*

Cao, Y., Li, H. & Petzold, L. (2004). Efficient formulation of the stochastic simulation algorithm for chemically reacting system, *Journal of Chemical Physics* **121**: 4059–4067.

Chang, L. & Karin, M. (2001). Mammalian MAP kinase signalling cascades, *Nature* **410**(6824): 37–40.
URL: *http://dx.doi.org/10.1038/35065000*

Csete, M. & Doyle, J. (2004). Bow ties, metabolism and disease, *Trends in Biotechnology* **22**(9): 446–450.

Elowitz, M. B., Levine, A. J., Siggia, E. D. & Swain, P. S. (2002). Stochastic gene expression in a single cell, *Science* **297**: 1183–1186.

Gally, D. L., Bogan, J. A., Eisenstein, B. I. & Blomfield, I. C. (1993). Environmental regulation of the *fim* switch controlling type 1 fimbrial phase variation in *Escherichia coli* K-12: effects of temperature and media, *J. Bacteriol.* **175**: 6186–6193.

Gillespie, D. T. (1976). A general method for numerically simulating the stochastic time evolution of coupled chemical reactions, *Journal of Computational Physics* **22**: 403–434.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions, *Journal of Physical Chemistry* **81**(25): 2340–2361.

Gillespie, D. T. (2000). The chemical Langevin equation, *Journal of Chemical Physics* **113**(1).

Gillespie, D. T. (2001). Approximate accelerated stochastic simulation of chemically reacting systems, *Journal of Chemical Physics* **115**(4): 1716–1733.

Gillespie, D. T. (2005). Stochastic chemical kinetics, *in* S. Yip (ed.), *Handbook of Materials Modeling*, Springer, pp. 1735–1752.

Gillespie, D. T. (2007). Stochastic simulation of chemical kinetics, *Annual Review of Physical Chemistry* **58**(1): 35–55.

Gillespie, D. T. & Petzold, L. R. (2003). Improved leap-size selection for accelerated stochastic simulation, *Journal of Chemical Physics* **119**.

Gillespie, D. T., Roh, M. & Petzold, L. R. (2009). Refining the weighted stochastic simulation algorithm, *The Journal of Chemical Physics* **130**(17): 174103.
URL: *http://link.aip.org/link/?JCP/130/174103/1*

Goldbeter, A. & Koshland, D. E. (1981). An amplified sensitivity arising from covalent modification in biological systems, *Proceedings of the National Academy of Sciences of the United States of America* **78**(11): 6840–6844.
URL: *http://www.pnas.org/content/78/11/6840.abstract*

Huang, C. Y. & Ferrell, J. E. (1996). Ultrasensitivity in the mitogen-activated protein kinase cascade, *Proceedings of the National Academy of Sciences of the United States of America* **93**(19): 10078–10083.
URL: *http://www.pnas.org/content/93/19/10078.abstract*

Istrail, S., De-Leon, S. B.-T. & Davidson, E. H. (2007). The regulatory genome and the computer, *Developmental Biology* **310**(2): 187 – 195.

Johnston, Jr., R. J. & Desplan, C. (2010). Stochastic mechanisms of cell fate specification that yield random or robust outcomes, *Annual Review of Cell and Developmental Biology* **26**(1).

URL: *https://www.annualreviews.orghttps://www.annualreviews.org/doi/abs/10.1146/annurev-cellbio-100109-104113*

Koller, D. & Friedman, N. (2009). *Probabilistic Graphical Models*, MIT Press.

Kuwahara, H. & Mura, I. (2008). An efficient and exact stochastic simulation method to analyze rare events in biochemical systems, *The Journal of Chemical Physics* **129**(16): 165101.
URL: *http://link.aip.org/link/?JCP/129/165101/1*

Kuwahara, H. & Myers, C. (2007). Production-passage-time approximation: A new approximation method to accelerate the simulation process of enzymatic reactions, *The 11th Annual International Conference on Research in Computational Molecular Biology*.

Kuwahara, H., Myers, C. J. & Samoilov, M. S. (2010). Temperature control of fimbriation circuit switch in uropathogenic *Escherichia coli*: Quantitative analysis via automated model abstraction, *PLoS Computational Biology* **6**(3): e1000723.

Kuwahara, H., Myers, C., Samoilov, M., Barker, N. & Arkin, A. (2006). Automated abstraction methodology for genetic regulatory networks, *Trans. on Comput. Syst. Biol.* **VI**: 150–175.

Little, J. W., Shepley, D. P. & Wert, D. W. (1999). Robustness of a gene regulatory circuit, *EMBO Journal* **18**: 4299–4307.

Losick, R. & Desplan, C. (2008). Stochasticity and Cell Fate, *Science* **320**(5872): 65–68.
URL: *http://www.sciencemag.org/cgi/content/abstract/320/5872/65*

Maheshri, N. & O'Shea, E. K. (2007). Living with noisy genes: How cells function reliably with inherent variability in gene expression, *Annual Review of Biophysics and Biomolecular Structure* **36**(1): 413–434.

McCollum, J. M., Peterson, G. D., Cox, C. D., Simpson, M. L. & Samatova, N. F. (2006). The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior, *Computational biology and chemistry* **30**(1): 39–49.

Misra, N. & Schwartz, R. (2008). Efficient stochastic sampling of first-passage times with applications to self-assembly simulations, *The Journal of Chemical Physics* **129**(20): 204109.

Murphy, K., Weiss, Y. & Jordan, M. (1999). Loopy belief propagation for approximate inference: An empirical study, *Uncertainty in AI*.

Nagel, L. W. & Pederson, D. (1973). Spice (simulation program with integrated circuit emphasis), *Technical Report UCB/ERL M382*, EECS Department, University of California, Berkeley.
URL: *http://www.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html*

Ptashne, M. (1992). *A Genetic Switch*, Cell Press & Blackwell Scientific Publishing.

Raj, A. & van Oudenaarden, A. (2008). Nature, nurture, or chance: Stochastic gene expression and its consequences, *Cell* **135**(2): 216–226.

Raser, J. M. & O'Shea, E. K. (2004). Control of stochasticity in eukaryotic gene expression, *Science* **304**: 1811–1814.

Reder, C. (1988). Metabolic control theory: A structural approach, *Journal of Theoretical Biology* **135**(2): 175 – 201.
URL: *http://www.sciencedirect.com/science/article/B6WMD-4KYW436-3/2/deaa46117df4b026f815bca0af0cbfeb*

Samad, H. E., Khammash, M., Petzold, L. & Gillespie, D. (2005). Stochastic modelling of gene regulatory networks, *International Journal of Robust and Nonlinear Control* **15**: 691–711.

Samoilov, M., Plyasunov, S. & Arkin, A. P. (2005). Stochastic amplification and signaling in enzymatic futile cycles through noise-induced bistability with oscillations, *Proceedings of the National Academy of Sciences US* **102**(7): 2310–5.

Samoilov, M. S. & Arkin, A. P. (2006). Deviant effects in molecular reaction pathways, *Nature Biotechnology* **24**: 1235–1240.

Sauro, H. M. & Ingalls, B. (2004). Conservation analysis in biochemical networks: computational issues for software writers., *Biophysical chemistry* **109**(1): 1–15.
URL: *http://www.ncbi.nlm.nih.gov/pubmed/15059656*

Schuster, S., Pfeiffer, T., Moldenhauer, F., Koch, I. & Dandekar, T. (2002). Exploring the pathway structure of metabolism: decomposition into subnetworks and application to *Mycoplasma pneumoniae* , *Bioinformatics* **18**(2): 351–361.
URL: *http://bioinformatics.oxfordjournals.org/cgi/content/abstract/18/2/351*

Vilar, J. M. G., Kueh, H. Y., Barkai, N. & Leibler, S. (2002). Mechanisms of noise-resistance in genetic oscillators, *Proceedings of the National Academy of Sciences of the US* **99**(9): 5988–5992.
URL: *http://www.pnas.org/content/99/9/5988.abstract*

Voet, D., Voet, J. & Pratt, C. (1999). *Fundamentals of biochemistry*, Wiley New York.

Wernet, M. F., Mazzoni, E. O., Celik, A., Duncan, D. M., Duncan, I. & Desplan, C. (2006). Stochastic spineless expression creates the retinal mosaic for colour vision, *Nature* **440**(7081): 174–180.
URL: *http://dx.doi.org/10.1038/nature04615*

Winstead, C., Madsen, C. & Myers, C. (2010). iSSA: an incremental stochastic simulation algorithm for genetic circuits, *Proc. 2010 IEEE International Symposium on Circuits and Systems (ISCAS 2010)*.

IntechOpen

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen