

Efficient exact and *K*-skip methods for stochastic simulation of coupled chemical reactions

Xiaodong Cai^{a)} and Ji Wen

Department of Electrical and Computer Engineering, University of Miami, Coral Gables, Florida 33124, USA

(Received 27 February 2009; accepted 22 July 2009; published online 12 August 2009)

Gillespie's direct method (DM) [D. Gillespie, *J. Chem. Phys.* **81**, 2340 (1977)] for exact stochastic simulation of chemical reaction systems has been widely adopted. It is easy to implement but requires large computation for relatively large systems. Recently, two more efficient methods, next reaction method (NRM) [M. A. Gibson and J. Bruck, *J. Phys. Chem. A* **105**, 1876 (2000)] and optimized DM (ODM) [Y. Cao *et al.*, *J. Chem. Phys.* **121**, 4059 (2004)], have been developed to improve simulation speed. It has been demonstrated that the ODM is the state-of-the-art most efficient method for exact stochastic simulation of most practical reaction systems. In this paper, we first develop an exact stochastic simulation algorithm named ODMK that is more efficient than the ODM. We then develop an approximate method named *K*-skip method to further accelerate simulation. Using two chemical reaction systems, we demonstrate that our ODMK and *K*-skip method can save 20%–30% and 70%–80% simulation time, respectively, comparing to the ODM. We also show that our ODMK and *K*-skip method provide almost the same simulation accuracy as the ODM. © 2009 American Institute of Physics. [DOI: [10.1063/1.3204422](https://doi.org/10.1063/1.3204422)]

I. INTRODUCTION

In certain chemical reaction systems such as cells in living organisms, the dynamics of the system are often dominated by the action of a small number of molecules, and thus exhibit significant stochastic fluctuations. Such random fluctuations in the number of molecules appear to have many important consequences in biology.^{1–3} For example, the stochasticity in gene expression can generate phenotypic heterogeneity in a bacterial population of isogenic cells.¹ Therefore, in computational modeling and simulation of chemical reaction systems, it is important to have an appropriate method to reflect the stochasticity in system dynamics. Gillespie developed a stochastic simulation algorithm (SSA) to simulate every reaction event in a chemical reaction system, when the time evolves.^{4,5} Gillespie's SSA is exact in the sense that it is developed rigorously based upon the microphysical premise of stochastic chemical kinetics for well-stirred chemical reaction systems.^{5,6}

Although Gillespie's exact SSA is easy to implement and produces system dynamics with correct statistics, it requires large computation especially for systems where the number of reaction channels and/or the number of molecules are large. A more efficient exact stochastic simulation method, named next reaction method (NRM), was developed by Gibson and Bruck.⁷ In the next reaction method, two sophisticated data structures are employed to save computation, thereby improving simulation speed. An optimized direct method (ODM) for exact stochastic simulation was proposed by Cao *et al.*⁸ It was demonstrated that the ODM is much faster than Gillespie's direct method (DM) and, in general,

faster than the NRM in simulating practical reaction systems. Several approximate leap methods, including τ -leap,^{9,10} binomial τ -leap,^{11,12} multinomial τ -leap,¹³ unbiased τ -leap,¹⁴ and *K*-leap methods,^{15,16} were also developed to accelerate stochastic simulation at the price of reduced simulation accuracy.

In this paper, we develop an exact SSA, named ODMK, which is more efficient than the ODM. In the DM and the ODM, two random variables are generated to simulate the occurrence of each reaction, one for the reaction index and the other for the random time between the occurrences of two consecutive reactions. In our ODMK, we use only one random variable to generate a sequence of $K > 1$ reaction indices. Since the number of random variables generated is reduced, our exact SSA is faster than existing exact SSAs. We further develop an approximate method, named *K*-skip method, where we generate the random time between the occurrence of every $K > 1$ reactions, skipping generation of the random time between the occurrence of two consecutive reactions. Since the random variables generated in our *K*-skip method is two per K reactions, which is much smaller than two per reaction in the DM and the ODM, our *K*-skip can significantly improve simulation speed especially when K is large. Our numerical results demonstrate that our *K*-skip method yields almost the same accuracy as the exact SSA.

The rest of this paper is organized as follows. In Sec. II, we describe the chemical reaction system under consideration and briefly review current exact SSAs. In Sec. III, we develop our ODMK. In Sec. IV, we derive our *K*-skip method. In Sec. V, we present two numerical examples to illustrate our algorithms, and compare their performance with that of the ODM. Finally, conclusions are drawn in Sec. VI.

^{a)}Author to whom correspondence should be addressed. Electronic mail: x.cai@miami.edu. Tel./Fax: (305) 284-5329/4044.

II. SYSTEM DESCRIPTION AND STOCHASTIC SIMULATION

Consider a well-stirred mixture of $N \geq 1$ molecular species $\{S_1, \dots, S_N\}$ that chemically interact through $M \geq 1$ reaction channels $\{R_1, \dots, R_M\}$. The dynamic state of this chemical system can be described by the state vector $\mathbf{X}(t) = [X_1(t), \dots, X_N(t)]^T$, where $X_n(t)$, $n=1, \dots, N$, is the number of S_n molecules at time t , and $[\cdot]^T$ denotes the transpose of the vector in the bracket. Following Gillespie,^{5,9,10} we define the dynamics of reaction R_m by a state-change vector $\mathbf{v}_m = [\nu_{1m}, \dots, \nu_{Nm}]^T$, where ν_{nm} gives the changes in the S_n molecular population produced by one R_m reaction, and a propensity function $a_m(\mathbf{x})$ together with the fundamental premise of stochastic chemical kinetics:

$$a_m(\mathbf{x})dt = \text{the probability, given } \mathbf{X}(t) = \mathbf{x}, \text{ that one reaction } R_m \text{ will occur in the next infinitesimal time interval } [t, t + dt). \quad (1)$$

Defining the probability rate constant c_m as the probability that a randomly selected combination of R_m reactant molecules react in a unit time period, we can calculate $a_m(\mathbf{x})$ as $a_m(\mathbf{x}) = c_m h_m(\mathbf{x})$,⁵ where $h_m(\mathbf{x})$ is the number of distinct combinations of R_m reactant molecules in the system at time t .

For a chemical system in a given state $\mathbf{X}(t) = \mathbf{x}$ at time t , Gillespie's exact SSA answers the following two questions:⁴ (1) when will the next reaction occur? and (2) which reaction will occur? More specifically, Gillespie's exact SSA simulates the following event in each step:

$$E: \text{no reaction occurs in the time interval } [t, t + \tau), \text{ and a reaction } R_\mu \text{ occurs in the infinitesimal time interval } [t + \tau, t + \tau + d\tau). \quad (2)$$

Based upon the fundamental premise (1), Gillespie showed that τ and μ are two independent random variables and have the following probability density functions (PDFs), respectively,^{4,5}

$$p_\tau(\tau) = a_0(\mathbf{x}) \exp(-a_0(\mathbf{x})\tau), \quad \tau > 0, \quad (3)$$

and

$$p_\mu(\mu) = \frac{a_\mu(\mathbf{x})}{a_0(\mathbf{x})}, \quad \mu = 1, \dots, M, \quad (4)$$

where $a_0(\mathbf{x}) = \sum_{m=1}^M a_m(\mathbf{x})$, and we used the same symbol to denote a random variable and its realization for notational simplicity. According to PDF (4), a realization of μ can be generated from a standard uniform random variable u_1 , by taking μ to be the integer that satisfies

$$\sum_{m=1}^{\mu-1} p_\mu(m) < u_1 \leq \sum_{m=1}^{\mu} p_\mu(m), \quad (5)$$

where we have defined $\sum_{m=1}^0 p_\mu(m) = 0$. Based on PDF (3), a realization of τ can be generated from another standard uniform random variable u_2 as $\tau = -\ln(u_2)/a_0(\mathbf{x})$. Therefore, Gillespie's DM generates a realization of μ and τ in each

step of the simulation, and then updates the time and system state as $t \leftarrow t + \tau$ and $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}_\mu$, respectively.

Gillespie also proposed a first reaction method (FRM) to simulate the event (2), but the FRM is much less efficient than the DM because it needs to generate more random variables. However, Gibson and Bruck⁷ transformed the FRM into an equivalent and more efficient NRM. In the NRM, a data structure named dependency graph is defined to tell precisely which $a_m(\mathbf{x})$ should be updated after a reaction occurs. An indexed priority queue is also defined to efficiently generate μ and τ . After employing these two data structures, it is argued that the NRM is more efficient than the DM,⁷ for loosely coupled chemical reaction systems where the firing of one reaction channel does not affect many other reactions.

However, a detailed analysis of CPU cost of both NRM and DM shows that maintaining and updating the indexed priority queue in the NRM may require significantly large cost for some practical systems.⁸ The ODM was proposed to improve the efficiency of the DM.⁸ The ODM incorporates the dependency graph used in the NRM into the DM to reduce the cost of calculating $a_m(\mathbf{x})$. It also properly orders the indices of reaction channels to reduce the cost of generating the reaction index μ . With these two optimization procedures, the ODM is much more efficient than the original DM. It was argued⁸ that the ODM is faster than the NRM in simulating most practical systems where some reactions fire more frequently than others.

III. EFFICIENT EXACT SSA

As we discussed in Sec. II, DM and ODM need to generate two uniform random variables to simulate the occurrence of each reaction: One for generating a reaction index μ and the other for generating a random time τ . Our idea for improving the efficiency of DM and ODM is to use one uniform random variable to generate a sequence of $K > 1$ reaction indices $\mu_1, \mu_2, \dots, \mu_K$. Recall that in DM and ODM, since we know the probability distribution of μ , $p_\mu(\mu)$ in Eq. (4), given the system state $\mathbf{X}(t) = \mathbf{x}$, we can easily generate a reaction index μ for the first reaction occurring after time t . Similarly, if we know the joint probability of K reaction indices $\mu_1, \mu_2, \dots, \mu_K$, $p(\mu_1, \mu_2, \dots, \mu_K)$, for the K reactions occurring sequentially after time t , we can use one uniform random variable to generate the sequence $\mu_1, \mu_2, \dots, \mu_K$. More specifically, for a specific sequence μ_1, \dots, μ_K , we define a vector

$$\boldsymbol{\mu}_l = [\mu_1, \mu_2, \dots, \mu_K], \quad l = \sum_{i=1}^K M^{K-i} (\mu_i - 1) + 1. \quad (6)$$

Note that since l is determined by μ_1, \dots, μ_K , different sequences of μ_1, \dots, μ_K have different value of l . Therefore, each $\boldsymbol{\mu}_l$ is a unique vector. We also define the joint cumulative distribution of K reaction indexes as $F(\boldsymbol{\mu}_l) = \sum_{i=1}^l p(\boldsymbol{\mu}_i)$. Using these definitions, we can first generate a uniform random variable u_1 and then find $\boldsymbol{\mu}_l$ by searching for $l \in \{1, 2, \dots, M^K\}$ that satisfies the following condition:

$$F(\boldsymbol{\mu}_{l-1}) < u_1 \leq F(\boldsymbol{\mu}_l), \quad (7)$$

where we have defined $F(\boldsymbol{\mu}_0)=0$. The joint probability $p(\boldsymbol{\mu}_l)$ can be written as

$$\begin{aligned} p(\boldsymbol{\mu}_l) &= p_{\mu_1}(\mu_1) \prod_{i=2}^K p_{\mu_i}(\mu_i | \mu_1, \dots, \mu_{i-1}) \\ &= p_{\mu_1}(\mu_1) \prod_{i=2}^K p_{\mu_i}(\mu_i | \mu_{i-1}), \end{aligned} \quad (8)$$

where $p_{\mu_1}(\mu_1)$ is the marginal probability distribution of μ_1 and $p_{\mu_i}(\mu_i | \mu_{i-1})$ is the conditional marginal distribution of μ_i given μ_{i-1} . Since $p_{\mu_1}(\mu_1)$ is known and $p_{\mu_i}(\mu_i | \mu_{i-1})$ can also be calculated from updated propensity functions after μ_1, \dots, μ_{i-1} have been specified, $p(\boldsymbol{\mu}_l)$ can be calculated for each $\boldsymbol{\mu}_l$, $l \in \{1, \dots, M^K\}$. However, the computation needed for calculating all $p(\boldsymbol{\mu}_l)$, $l \in \{1, \dots, M^K\}$, increases exponentially with K . Moreover, even after we calculate all $p(\boldsymbol{\mu}_l)$, it seems that searching for $\boldsymbol{\mu}_l$ over a set of M^K elements requires a complexity again increasing exponentially with K . It turns out that these two problems can be overcome. In the following, we derive a method for finding $\boldsymbol{\mu}_l$ that satisfies Eq. (7) with a linear complexity with respect to K .

Specifically, let us define a variable \tilde{u}_i , $i=1, \dots, K$, as follows:

$$\tilde{u}_i = \begin{cases} u_1, & i=1, \\ \frac{u_1 - \sum_{m=1}^{\mu_1-1} p_{\mu_1}(m)}{p_{\mu_1}(\mu_1)}, & i=2, \\ \frac{\tilde{u}_{i-1} - \sum_{m=1}^{\mu_{i-1}-1} p_{\mu_{i-1}}(m | \mu_{i-2})}{p_{\mu_{i-1}}(\mu_{i-1} | \mu_{i-2})}, & i > 2. \end{cases} \quad (9)$$

In Appendix A, we prove the following proposition:

Proposition 1. *Given the definition of $\boldsymbol{\mu}_l$ in Eq. (6), condition (7) is satisfied if and only if the following conditions hold true*

$$\sum_{m_1=1}^{\mu_1-1} p_{\mu_1}(m_1) < \tilde{u}_1 \leq \sum_{m_1=1}^{\mu_1} p_{\mu_1}(m_1) \quad (10)$$

$$\sum_{m_i=1}^{\mu_i-1} p_{\mu_i}(m_i | \mu_{i-1}) < \tilde{u}_i \leq \sum_{m_i=1}^{\mu_i} p_{\mu_i}(m_i | \mu_{i-1}), \quad i=2, \dots, K.$$

Therefore, instead of using Eq. (7) to search for the sequence μ_1, \dots, μ_K jointly in a set of M^K elements, we can use Eq. (10) to search for each of μ_1, \dots, μ_K sequentially in a set of M elements. Comparing Eq. (10) with Eq. (5), we see that the complexity of searching for reaction indices using Eq. (10) is the same as that using Eq. (5) in the DM, but we can use a single uniform random variable for a sequence of K reaction indices. Since μ_1, \dots, μ_K are generated exactly from their joint distribution (8) and the marginal distribution of μ_k , $k=1, \dots, K$, specified in Eq. (8) is identical to Eq. (4) used in Gillespie's exact SSA, reaction indices generated from our method have the same statistics as those generated from Gillespie's exact SSA; and apparently, our method does not incur any bias in the generation of reaction indices.

We refer to our exact SSA that uses Eq. (10) to generate reaction indices as DMK and summarize it as follows:

Algorithm 1. (DMK)

1. Initialization (set $t=0$ and the initial number of molecules $\mathbf{X}(t)=\mathbf{x}$, calculate propensity functions $a_m(\mathbf{x})$, $m=1, \dots, M$, $a_0(\mathbf{x})=\sum_{m=1}^M a_m(\mathbf{x})$, and choose K).

2. Generate a standard uniform random variable u_1 , set $k=1$.

3. Find μ such that

$$\frac{\sum_{m=1}^{\mu-1} a_m(\mathbf{x})}{a_0(\mathbf{x})} < u_1 \leq \frac{\sum_{m=1}^{\mu} a_m(\mathbf{x})}{a_0(\mathbf{x})}.$$

4. If $k < K$, set

$$u_1 \leftarrow \frac{u_1 - \sum_{m=1}^{\mu-1} a_m(\mathbf{x})/a_0(\mathbf{x})}{a_{\mu}(\mathbf{x})/a_0(\mathbf{x})}.$$

Set $k \leftarrow k+1$.

5. Generate a standard uniform random variable u_2 and calculate $\tau = -\ln(u_2)/a_0(\mathbf{x})$.

6. Update the time as $t \leftarrow t + \tau$ and the state vector as $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}_{\mu}$. Calculate propensity functions $a_m(\mathbf{x})$, $m=1, \dots, M$ and $a_0(\mathbf{x})=\sum_{m=1}^M a_m(\mathbf{x})$.

7. If t is equal to or greater than the end time, stop; otherwise, if $k \leq K$ go to step 3 or else go to step 2.

Note that the two optimization procedures employed in the ODM can also be incorporated into Algorithm 1. We refer to the optimized version of Algorithm 1 as ODMK. The differences between ODMK and ODM are as follows: (1) to simulate the occurrence of each reaction, the ODM generates two uniform random variables, whereas our ODMK generates $1+1/K$ uniform random variables on average, and (2) our ODMK needs to calculate u_1 in step 4, whereas the ODM does not have this step. Since calculation of u_1 only needs one extra subtraction and two extra divisions, its computational complexity is much lower than that of a good uniform random variable generator, such as run2 in Ref. 18. Therefore, our ODMK is more efficient than the ODM. Clearly, increasing K will reduce computation, but when K is greater than certain value, say 100, the reduction in computation is negligible if further increasing K . Generally, we can select a value around 100 for K .

IV. K-SKIP SSA

In exact SSAs, we need to generate a random time for the occurrence of each reaction. Consider K reactions that occur sequentially after t . Exact SSAs generate K random times, say τ_1, \dots, τ_K , each accounting for a random time that the system waits before the next reaction occurs. If we only generate $\tau = \sum_{k=1}^K \tau_k$, skipping the generation of each individual time, and update the system time after K reactions occur, we can save more computational time. This is in fact the idea behind our K -Skip methods that we will describe next.

The time τ_k has an exponential distribution with a parameter $a_0(\mathbf{x}_k)$ as described in Eq. (3), where \mathbf{x}_k is the system state before the k th reaction occurs. Since $\tau_k, k=1, \dots, K$, are

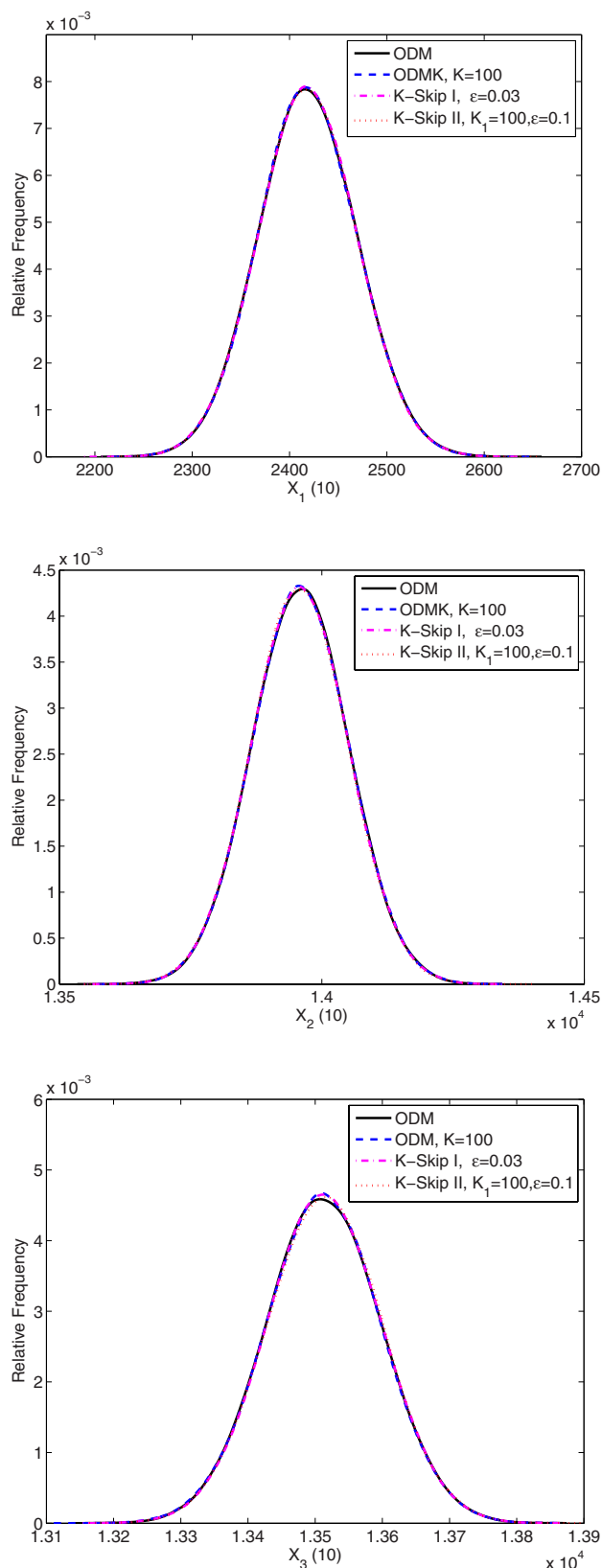


FIG. 1. Histograms of the number of molecules for the decaying-dimerizing reactions given in Eq. (15). The histograms were obtained from 10^5 simulation runs.

independent, the probability distribution of $\tau = \sum_{k=1}^K \tau_k$ can be found in a closed form.¹⁷ When $a_0(x_k)$, $k=1, \dots, K$ are all equal, τ follows a Gamma distribution with parameters K and $a = a_0(x_k)$:

$$p(\tau) = \frac{a \exp(-a\tau)(a\tau)^{K-1}}{(K-1)!}, \quad \tau > 0. \quad (11)$$

However, the distribution of τ is in a complicated form, when all $a_0(x_k)$'s are different or when some $a_0(x_k)$'s are equal but the others are different, which is often true when the system evolves and the system state changes constantly. In these cases, it is impossible to derive an inverse function for the cumulative distribution function (CDF) of τ and difficult to find a proper upper bound for the PDF of τ . Therefore, it is difficult to generate τ from its probability distribution directly or using a rejection method. However, since the occurrence of one reaction only changes the system state slightly, it is expected that one reaction causes a small change in $a_0(x)$. If we choose K carefully, we can ensure that $a_0(x_k)$, $k=1, \dots, K$, are approximately equal, and thus we can approximate the distribution of τ by a Gamma distribution and easily generate τ from the Gamma distribution.

More specifically, we impose the following condition to ensure that $a_0(x_k)$, $k=1, \dots, K$, are approximately equal:

$$\max_{i,j \in \{1, \dots, K\}} |a_0(x_i) - a_0(x_j)| < \epsilon a_0(x_1), \quad (12)$$

where ϵ is a prespecified small positive constant. Since calculating the left hand side of Eq. (12) needs relatively large computation, we simplify Eq. (12) as follows:

$$|a_0(x_k) - a_0(x_1)| < \frac{\epsilon a_0(x_1)}{2}, \quad \forall k \leq K. \quad (13)$$

In simulation, we can increase K so long as Eq. (13) is satisfied and take final K as the maximum value that satisfies Eq. (13). The mean value of τ is $E[\tau] = \sum_{k=1}^K 1/a_0(x_k)$. If τ follows a Gamma distribution given in Eq. (11), its mean is K/a . Therefore, once K is determined, we can generate τ from a Gamma distribution with parameters K and

$$a = \frac{K}{\sum_{k=1}^K 1/a_0(x_k)}, \quad (14)$$

which ensures that the approximate τ is unbiased.

In summary, if we generate μ_1, \dots, μ_K from their exact joint distribution as in Algorithm 1, but generate τ from its approximate Gamma distribution as discussed earlier, we get an approximate but faster algorithm. We refer to this algorithm as K -skip method I, since we will also develop another method, referred to as K -skip method II, to further improve simulation speed by generating μ_1, \dots, μ_K from an approximate distribution. K -skip method I is summarized in the following algorithm:

Algorithm 2. (K -skip method I)

1. Initialization (set $t=0$ and the initial number of molecules $\mathbf{X}(t) = \mathbf{x}$, calculate propensity functions $a_m(\mathbf{x})$, $m = 1, \dots, M$, $a_0(\mathbf{x}) = \sum_{m=1}^M a_m(\mathbf{x})$ and choose ϵ).

2. Generate a standard uniform random variable u_1 , set $\tilde{a}_0 = a_0(\mathbf{x})$, $b = \epsilon a_0(\mathbf{x})/2$, $c = 1/a_0(\mathbf{x})$ and $K=0$.

3. Find μ such that

$$\frac{\sum_{m=1}^{\mu-1} a_m(\mathbf{x})}{a_0(\mathbf{x})} < u_1 \leq \frac{\sum_{m=1}^{\mu} a_m(\mathbf{x})}{a_0(\mathbf{x})}.$$

TABLE I. Histogram distances for the decaying-dimerizing reactions.

	ODM vs ODM	ODMK vs ODM			K-Skip I vs ODM			K-Skip II vs ODM
	Mean \pm Std.	$K=10$	$K=100$	$K=1000$	$\epsilon=0.01$	$\epsilon=0.03$	$\epsilon=0.1$	$K=100, \epsilon=0.1$
X_1	0.0094 \pm 0.0023	0.0105	0.0088	0.0100	0.0093	0.0091	0.0097	0.0088
X_2	0.0092 \pm 0.0019	0.0077	0.0085	0.0079	0.0082	0.0097	0.0077	0.0183
X_3	0.0092 \pm 0.0021	0.0095	0.0112	0.0091	0.0100	0.0105	0.0083	0.0179

4. Update the state vector as $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}_\mu$, calculate propensity functions $a_m(\mathbf{x})$, $m=1, \dots, M$, and $a_0(\mathbf{x}) = \sum_{m=1}^M a_m(\mathbf{x})$, and set $K \leftarrow K+1$.

5. If $|a_0(\mathbf{x}) - \tilde{a}_0| < b$, update $c \leftarrow c + 1/a_0(\mathbf{x})$ and update u_1 as follows

$$u_1 \leftarrow \frac{u_1 - \sum_{m=1}^{\mu-1} a_m(\mathbf{x})/a_0(\mathbf{x})}{a_\mu(\mathbf{x})/a_0(\mathbf{x})},$$

and then go to step 3; otherwise, go to step 6.

6. Generate τ from a Gamma distribution with parameters K and $a=K/c$.

7. Update the time as $t \leftarrow t + \tau$. If t is equal to or greater than the end time, stop; otherwise, go to step 2.

Algorithm 2 can also be optimized using the two optimization procedures employed in the ODM. The differences between Algorithm 2 and Algorithm 1 are as follows: (1) Algorithm 2 generates one Gamma random variable every K reactions, whereas Algorithm 1 generates K exponential random variables for K reactions, (2) Algorithm 2 needs to calculate b in step 2, c in steps 2 and 5, a in step 6, and do comparison $|a_0(\mathbf{x}) - \tilde{a}_0| < b$ in step 5, whereas Algorithm 1 does not need such computation. A Gamma random variable can be generated efficiently.¹⁸ The computation saved by generating only one Gamma random variable rather than K exponential random variables significantly outweighs the extra computation needed in Algorithm 2, especially when K is relatively large. Therefore, Algorithm 2 improves simulation speed. Note that reaction index for each reaction is generated using its exact distribution. The only approximation error comes from τ , which is controlled by parameter ϵ .

Algorithm 2 and exact SSAs spend much computation on generating a reaction index μ and updating the propensity functions after a reaction index is generated. As we mentioned earlier, it is reasonably expected that the occurrence of one reaction only changes propensity functions slightly. Therefore, we can update propensity functions after occurrence of $1 < K_1 \leq K$ reactions, where K_1 can be chosen to ensure small changes in propensity functions after K_1 reactions occur. More specifically, we can modify Algorithm 2 to further improve simulation efficiency as follows: (1) Update

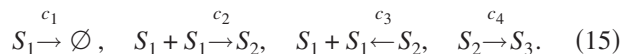
state vector and calculate propensity functions every K_1 reactions in step 4, and (2) calculate the lower and upper limits of u_1 in step 3 every K_1 reactions. We refer to the algorithm incorporating these two modifications as K -skip method II. Note that K -skip method II improve simulation efficiency at the price of increasing simulation errors, since reaction indices are no longer generated from their exact probability distribution as in Algorithms 1 and 2. However, we can choose K_1 to ensure that the approximation error is small.

V. NUMERICAL EXAMPLES

In this section, we present simulation results for two chemical reaction systems to demonstrate the accuracy and efficiency of our ODMK and K -skip method. We ran simulations using the ODM, our ODMK and K -skip method which was also optimized using the same two optimizations procedures used in the ODM. We employed histogram distance¹⁹ between the simulation results of the ODM and those of our methods to measure the simulation accuracy. For molecular species with a small number of molecules, the frequency of each number of molecules was calculated as histogram; for molecular species with a large number of molecules, we used the Parzen method²⁰ with a Gaussian window to estimate the histogram. We used algorithm ran2 in Ref. 18 for generating uniform random variable. In the K -skip method, we employed the Gamma random number generator in Ref. 18. All simulation programs were written in C, and run on a PC with a 3.2 GHz CPU and 2 Gbyte memory running Windows XP.

A. Decaying-dimerizing reactions

This example was originally used by Gillespie^{9,10} to test his SSAs. It includes the following four reactions:



In these reactions, a monomer S_1 degrades by itself and also reversibly dimerizes to an unstable form S_2 , which can convert to a stable form S_3 . We simulate these reactions using

TABLE II. CPU time (in seconds) of 10^5 simulation runs and speedup over the ODM for the decaying-dimerizing reactions.

	ODM	ODMK			K-Skip I			K-Skip II
		$K=10$	$K=100$	$K=1000$	$\epsilon=0.01$	$\epsilon=0.03$	$\epsilon=0.1$	$K=100, \epsilon=0.1$
Time	5209.7	3990.1	3821.9	3805.8	1997.4	1809.2	1782.6	1324.0
Speedup	1.00	1.31	1.36	1.37	2.61	2.88	2.92	3.93

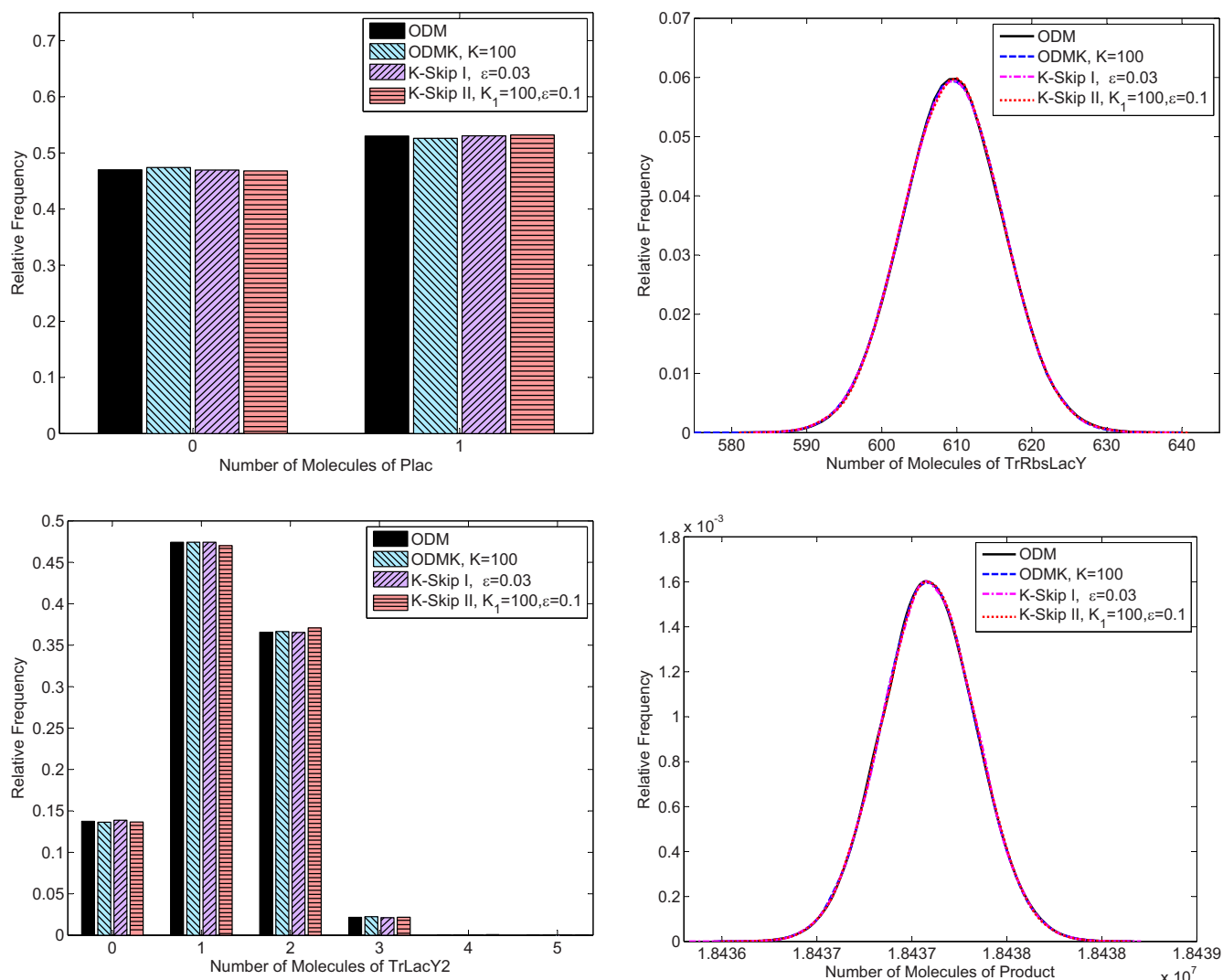


FIG. 2. Histograms of the number of molecules for the LacZ/LacY reactions. The histograms were obtained from 10^5 simulation runs.

the same values of rate constant and initial conditions as Gillespie¹⁰

$$c_1 = 1, \quad c_2 = 0.002, \quad c_3 = 0.5, \quad c_4 = 0.04, \quad (16)$$

and

$$X_1(0) = 4150, \quad X_2(0) = 39\,565, \quad X_3(0) = 3445. \quad (17)$$

We run simulation 10^5 times, each time starting at $t=0$ and ending at $t=10$.

Figure 1 depicts the histograms of $X_1(10)$, $X_2(10)$, and $X_3(10)$ obtained from 10^5 simulation runs. It is seen that our ODMK has a histogram almost identical to that of the ODM, as expected, since both are exact SSAs. The histograms of two K -skip methods are almost the same as those of exact SSAs, which implies that simulation errors of K -skip methods are negligible. Table I lists the histogram distance between the simulation results obtain from the ODM and our methods. We also give the mean and standard deviation of the self-histogram distance of the ODM obtained as follows. We ran simulations 10^5 times using the ODM and got a histogram for each molecular species, and then repeated this process 50 times and got a total of 50 histograms for each species. Using these 50 histograms, we calculated $\binom{50}{2}$

= 1225 histogram distances and then calculated the mean and standard deviation of these 1225 histogram distances. The mean self-histogram distance of the ODM is about 0.01. It is seen that all the histogram distances between the ODM and our methods are very small: Five of them (highlighted in italic font) are less than 0.02 and the others are less than 0.01. This shows that simulation errors of K -skip methods are negligible in this particular example. Table II lists the CPU time of 10^5 simulation runs for the ODM and our methods. We also gives the speedup of our method relative to the ODM defined as follows:

$$\text{speedup} = \frac{\text{CPU time of the ODM}}{\text{CPU time of our method}}. \quad (18)$$

The speedup of our ODMK is 1.3. The speedup of K -leap method I depends on ϵ and is about 3 for $\epsilon \geq 0.03$. The speedup of K -leap method II depends on ϵ and K_1 and is about 4 for $\epsilon=0.1$ and $K_1=100$. Note that although the speedup of K -skip methods is not as large as that of τ -leap method,^{9,10} binomial τ -leap method^{11,12} or K -leap method,^{15,16} the histogram distance of our K -skip methods is negligible, while leap methods generally have a considerably

TABLE III. Histogram distances for the LacZ/LacY system.

	ODM vs ODM	ODMK vs ODM			K-Skip I vs ODM			K-Skip II vs ODM
	Mean \pm Std.	$K=10$	$K=100$	$K=1000$	$\epsilon=0.01$	$\epsilon=0.03$	$\epsilon=0.1$	$K=100, \epsilon=0.1$
Plac	0.0034 \pm 0.0027	0.0004	0.0081	0.0025	0.0001	0.0009	0.0006	0.0040
RNAP	0.0058 \pm 0.0024	0.0052	0.0038	0.0062	0.0056	0.0066	0.0089	0.0087
PLacRNAP	0.0035 \pm 0.0029	0.0007	0.0032	0.0000	0.0009	0.0028	0.0012	0.0041
TrLacZ1	0.0025 \pm 0.0021	0.0003	0.0050	0.0025	0.0008	0.0019	0.0006	0.0001
RbsLacZ	0.0056 \pm 0.0025	0.0028	0.0028	0.0052	0.0026	0.0068	0.0043	0.0066
TrLacZ2	0.0042 \pm 0.0023	0.0056	0.0022	0.0033	0.0050	0.0016	0.0018	0.0028
TrLacY1	0.0040 \pm 0.0025	0.0028	0.0057	0.0024	0.0069	0.0030	0.0076	<i>0.0126</i>
RbsLacY	0.0064 \pm 0.0026	0.0070	0.0062	0.0051	0.0088	0.0053	0.0061	0.0033
TrLacY2	0.0048 \pm 0.0022	0.0023	0.0028	0.0009	0.0030	0.0024	0.0066	<i>0.0103</i>
Ribosome	0.0048 \pm 0.0015	0.0040	0.0036	0.0085	0.0052	0.0041	0.0059	0.0502
RbsRibosomeLacZ	0.0078 \pm 0.0028	0.0051	0.0097	0.0055	0.0039	0.0105	0.0080	<i>0.1289</i>
RbsRibosomeLacY	0.0080 \pm 0.0025	0.0092	0.0058	0.0065	0.0056	0.0065	0.0061	<i>0.1326</i>
TrRbsLacZ	<i>0.0103</i> \pm 0.0023	0.0085	0.0071	<i>0.0132</i>	<i>0.0134</i>	0.0086	0.0100	0.0086
TrRbsLacY	0.0104 \pm 0.0022	0.0072	0.0085	0.0116	0.0095	0.0098	0.0092	0.0100
LacZ	0.0085 \pm 0.0019	<i>0.0107</i>	0.0054	0.0061	0.0070	0.0070	0.0077	0.0071
LacY	0.0085 \pm 0.0024	0.0079	0.0105	0.0106	0.0084	0.0093	0.0071	0.0073
Lactose	<i>0.0107</i> \pm 0.0023	0.0085	<i>0.0136</i>	0.0117	0.0113	0.0117	0.0099	<i>0.0133</i>
LacZlactose	0.0085 \pm 0.0018	0.0099	0.0073	0.0046	0.0093	0.0072	<i>0.0121</i>	0.0066
Product	<i>0.0103</i> \pm 0.0018	<i>0.0119</i>	0.0088	0.0105	0.0113	0.0118	<i>0.0166</i>	<i>0.0103</i>

large histogram distance. In fact, if the step size of a leap method is chosen so small as to make the histogram distance very small, the leap method may be slower than an exact SSA. It was suggested that when a step size in a leap method is small, simulation should be run using an exact SSA for the occurrence of hundreds of reactions and then attempting to use the leap method.^{9,21} Therefore, from the viewpoint of simulation speed and accuracy, our K -skip method fills the gap between the exact SSA and the τ - or K -leap method, since it provides negligible errors but relatively smaller speedup when comparing to the τ - or K -leap method.

B. Expression of LacZ/LacY genes and activities of LacZ/LacY proteins

This example contains 19 molecular species and 22 reaction channels, described in detail by Kierzek,²² and Tian and Burrage.¹¹ We first ran simulation using the ODM, starting at $t=0$ with initial condition given by Kierzek, and ending at $t=600$. We then used the result of the exact SSA as the initial condition, and ran simulation 10^5 times using our ODMK, K -skip methods, as well as the ODM. Each run started at $t=600$ and ended at $t=601$. Following Tian and Burrage,¹¹ we generated the initial number of RNAP molecules at $t=600$ from a Gaussian random variable with mean 35 and standard deviation 3.5, and the initial number of ribosome molecules $t=600$ from a Gaussian random variable with mean 350 and standard deviation 35.

Figure 2 depicts the histograms of four molecular species: Plac, Product, TrLacY2, and TrRbsLacY. Plac is the promoter of LacZ gene without being bound by an RNA polymerase and its number of molecules is either 0 or 1; TrLacY2 is the RNA polymerase that is transcribing the LacY gene and its number of molecules is relatively small ranging from 0 to 5. TrRbsLacY is the ribosome that is translating LacY mRNA and its number of molecules is relatively

large: Product is the product converted from lactose in a reaction catalyzed by LacZ protein and its number of molecules is very large. For all four species, the histograms of our ODMK and K -skip methods are almost the same as those of the ODM.

Table III lists the histogram distance between the simulation results obtain from the ODM and our methods. As in the previous example, we also give the mean and standard deviation of the self-histogram distance of the ODM. The mean self-distance of the ODM range from 0.0034 to 0.0107. Most of the histogram distances between our method and the ODM are less than 0.01; eleven histogram distances (highlighted in italic font) are between 0.01 and 0.02; only two histogram distances (highlighted in italic font) between our K -skip method II and ODM are relatively large: 0.1289 and 0.1326. This shows that simulation errors of K -skip method I are negligible in this example, and that the errors of K -skip method II are small but not negligible. Table IV lists the CPU time of 10^5 simulation runs for the ODM and our methods, as well as the speedup of our methods relative to the ODM. The speedup is about 1.4 for the ODMK, about 3 for K -skip method I and about 4 for K -skip method II.

VI. CONCLUSION

Gillespie's DM is a widely used method for exact stochastic simulation of chemical reaction systems, which simulates every reaction event by using two random variables: One for generating reaction index and the other for generating the time that the system needs to wait until the next reaction occurs. In this paper, we developed an exact SSA named DMK that is more efficient than Gillespie's DM. In our DMK, we use one random variable to generate a sequence of $K > 1$ reaction indices, and thus, we only need 1

TABLE IV. CPU time (in seconds) of 10^5 simulation runs and speedup over the ODM for the LacZ/LacY system.

	ODMK			K-Skip I			K-Skip II	
	ODM	$K=10$	$K=100$	$K=1000$	$\epsilon=0.01$	$\epsilon=0.03$	$\epsilon=0.1$	$K=100, \epsilon=0.1$
Time	5474.6	4109.4	3950.8	3941.2	2543.7	1813.4	1679.3	1319.2
Speedup	1.00	1.33	1.39	1.39	2.15	3.02	3.26	4.15

$+1/K$ random variables on average to simulate a reaction event, which improves simulation efficiency. Incorporating two optimization procedures employed in the ODM into our DMK, we converted DMK to an optimized method, named ODMK, which is more efficient than ODM.

Based on the ODMK, we also developed an approximate method, the K -skip method, where we used one random variable to generate the time needed for the occurrence of K reactions, skipping the time needed for the occurrence of each individual reaction. Since the K -skip method needs only two random variables to simulate K reaction events, it further improves simulation efficiency. Applying our ODMK and K -skip method to two reaction systems, we showed that our methods provided almost the same simulation accuracy as the ODM, but our ODMK and K -skip method could save 20%–30% and 70%–80% simulation time, respectively, relative to the ODM. The idea behind our ODMK and K -skip method can also be applied to the SSA for chemical reaction systems with time delay.²³

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (NSF) under NSF CAREER Award No. 0746882.

APPENDIX A: PROOF OF PROPOSITION 1

We first prove the “only if” part of the proposition. From the definition of μ_i in Eq. (6), the sequence of reaction indices in μ_i can be specified by nodes on a path from the root to a leaf in the tree described in Fig. 3, with μ_1 corresponding to the lowest path and μ_{MK} corresponding to the upmost path. Based on this tree structure, we can express $F(\mu_i)$ as follows:

$$\begin{aligned}
 F(\mu_i) &= \sum_{m_1=1}^{\mu_1} \sum_{m_2=1}^{\mu_2} \cdots \sum_{m_K=1}^{\mu_K} p(m_1, \dots, m_K) \\
 &= \sum_{m_1=1}^{\mu_1-1} \sum_{m_2=1}^M \cdots \sum_{m_K=1}^M p(m_1, \dots, m_K) \\
 &\quad + \sum_{m_2=1}^{\mu_2} \cdots \sum_{m_K=1}^{\mu_K} p(\mu_1, m_2, \dots, m_K) \\
 &= \sum_{m_1=1}^{\mu_1-1} p_{\mu_1}(m_1) + p_{\mu_1}(\mu_1) F_2(\mu_i), \quad (\text{A1})
 \end{aligned}$$

where we have defined $\sum_{m_1=1}^0 p_{\mu_1}(m_1) = 0$ and $F_2(\mu_i)$ is defined as

$$F_2(\mu_i) = \sum_{m_2=1}^{\mu_2} \cdots \sum_{m_K=1}^{\mu_K} p(m_2, \dots, m_K | \mu_i). \quad (\text{A2})$$

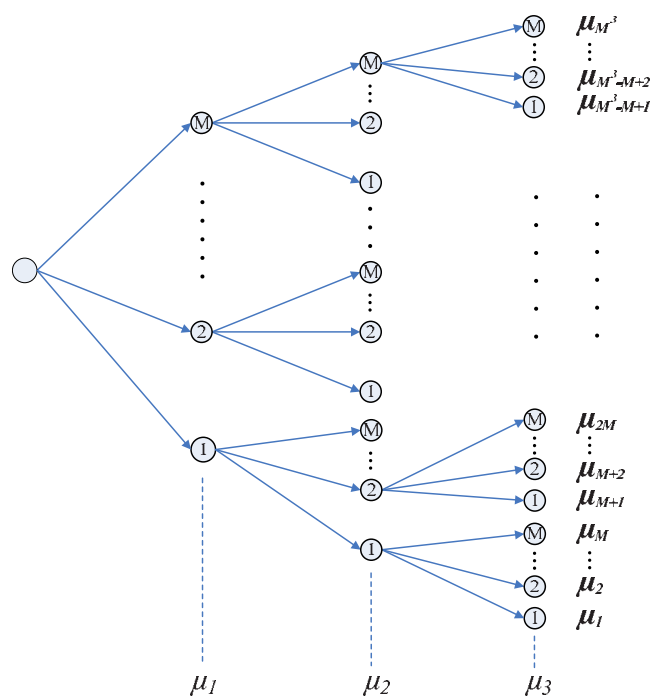
Since $F_2(\mu_i) \leq 1$, using Eq. (A1), we infer that if the second inequality in Eq. (7) is satisfied, we must have

$$u_1 \leq \sum_{m_1=1}^{\mu_1} p_{\mu_1}(m_1). \quad (\text{A3})$$

For the sequence $\mu_i = [\mu_1, \mu_2, \dots, \mu_K]$, if $\mu_i > 1$ and $\mu_{i+1} = \mu_{i+2} = \dots = \mu_K = 1$ for $1 \leq i \leq K$, then we have $\mu_{i-1} = [\mu_1, \dots, \mu_{i-1}, \mu_i - 1, M, \dots, M]$. Therefore, we can write $F(\mu_{i-1})$ as

$$\begin{aligned}
 F(\mu_{i-1}) &= \sum_{m_1=1}^{\mu_1} \cdots \sum_{m_{i-1}=1}^{\mu_{i-1}} \sum_{m_i=1}^{\mu_i-1} \sum_{m_{i+1}=1}^M \cdots \sum_{m_K=1}^M p(m_1, \dots, m_K) \\
 &= \sum_{m_1=1}^{\mu_1-1} p_{\mu_1}(m_1) + p_{\mu_1}(\mu_1) F_2(\mu_{i-1}), \quad (\text{A4})
 \end{aligned}$$

where we have defined $p_{\mu_1}(0) = 0$, and $F_2(\mu_{i-1})$ is defined as

FIG. 3. The tree structure of K reaction indices with $K=3$.

$$F_2(\boldsymbol{\mu}_{l-1}) = \begin{cases} 0, & i = 1, \\ \sum_{m_2=1}^{\mu_2} \cdots \sum_{m_{i-1}=1}^{\mu_{i-1}} \sum_{m_i=1}^{\mu_i-1} p(m_2, \dots, m_i | \boldsymbol{\mu}_1), & 1 < i \leq K. \end{cases} \quad (\text{A5})$$

Since $F_2(\boldsymbol{\mu}_{l-1}) \geq 0$, using Eq. (A4), we infer that if the first inequality in Eq. (7) is satisfied, we must have

$$u_1 > \sum_{m_1=1}^{\mu_1-1} p_{\mu_1}(m_1). \quad (\text{A6})$$

Combining Eqs. (A3) and (A6), we see that Eq. (7) together with Eq. (6) imply the first condition in Eq. (10) which is essentially the same as Eq. (5). Note that without the definition of $\boldsymbol{\mu}_l$ in Eq. (6) or the tree structure in Fig. 3, we may not be able to derive Eqs. (A3) and (A6) from Eq. (7).

If we define \tilde{u}_2 as that in Eq. (9), then Eqs. (7), (A1), and (A4) imply that

$$F_2(\boldsymbol{\mu}_{l-1}) < \tilde{u}_2 \leq F_2(\boldsymbol{\mu}_l). \quad (\text{A7})$$

Similar to the derivation of Eqs. (A3) and (A6) from Eq. (7), we can derive the following inequalities from Eq. (A7)

$$\sum_{m_2=1}^{\mu_2-1} p_{\mu_2}(m_2 | \boldsymbol{\mu}_1) < \tilde{u}_2 \leq \sum_{m_2=1}^{\mu_2} p_{\mu_2}(m_2 | \boldsymbol{\mu}_1). \quad (\text{A8})$$

If we define \tilde{u}_i , $i=1, \dots, K$, recursively, as in Eq. (9), then, similar to the derivation of Eq. (A8), we can derive the second condition in Eq. (10) for $i > 2$. This finishes the proof of the “only if” part of the proposition.

We now prove the “if” part of the proposition. Using Eq. (9), we can construct a sequence, μ_1, \dots, μ_K , from u_1 that satisfies Eq. (10) as follows. First, we find μ_1 that satisfies the two inequalities imposed on $\tilde{u}_1 = u_1$ in Eq. (10). After calculating \tilde{u}_2 from \tilde{u}_1 and μ_1 using Eq. (9), we find μ_2 that satisfies the two inequalities on \tilde{u}_2 in Eq. (10). After calculating \tilde{u}_3 from \tilde{u}_2 and μ_2 using Eq. (9), we then find μ_3 that satisfies the two inequalities on \tilde{u}_3 in Eq. (10). This process continues until we get μ_K . We next prove that the sequence such constructed is the sequence $\boldsymbol{\mu}_l$ that satisfies Eq. (7).

From Eq. (9), we have

$$u_1 = \tilde{u}_1 = \sum_{m=1}^{\mu_1-1} p_{\mu_1}(m) + \tilde{u}_2 p_{\mu_1}(\mu_1), \quad (\text{A9})$$

$$\tilde{u}_{i-1} = \sum_{m=1}^{\mu_{i-1}-1} p_{\mu_{i-1}}(m | \mu_{i-2}) + \tilde{u}_i p_{\mu_{i-1}}(\mu_{i-1} | \mu_{i-2}), \quad i \geq 3.$$

Using Eq. (A9), it is not difficult to express u_1 as follows:

$$u_1 = \sum_{k=1}^{K-1} \left\{ \left[\prod_{i=1}^{k-1} p_{\mu_i}(\mu_i | \mu_{i-1}) \right] \sum_{m=1}^{\mu_k-1} p_{\mu_k}(m | \mu_{k-1}) \right\} + \tilde{u}_K \prod_{i=1}^{K-1} p_{\mu_i}(\mu_i | \mu_{i-1}), \quad (\text{A10})$$

where we have defined $p_{\mu_1}(m | \mu_0) = p_{\mu_1}(m)$ and $\prod_{i=1}^0 p_{\mu_i}(\mu_i | \mu_{i-1}) = 1$.

If we continue the derivation in Eq. (A1) until replacing the joint distribution with marginal distributions, we can write $F(\boldsymbol{\mu}_l)$ as follows:

$$F(\boldsymbol{\mu}_l) = \sum_{k=1}^{K-1} \left\{ \left[\prod_{i=1}^{k-1} p_{\mu_i}(\mu_i | \mu_{i-1}) \right] \sum_{m=1}^{\mu_k-1} p_{\mu_k}(m | \mu_{k-1}) \right\} + \left[\sum_{m=1}^{\mu_K} p_{\mu_K}(m | \mu_{K-1}) \right] \prod_{i=1}^{K-1} p_{\mu_i}(\mu_i | \mu_{i-1}). \quad (\text{A11})$$

Comparing Eq. (A10) with Eq. (A11) and using the last inequality in Eq. (10), $\tilde{u}_K \leq \sum_{m=1}^{\mu_K} p_{\mu_K}(m | \mu_{K-1})$, we conclude that

$$u_1 \leq F(\boldsymbol{\mu}_l). \quad (\text{A12})$$

Using Eqs. (A11) and (8), we have

$$F(\boldsymbol{\mu}_{l-1}) = F(\boldsymbol{\mu}_l) - p(\boldsymbol{\mu}_l) = \sum_{k=1}^{K-1} \left\{ \left[\prod_{i=1}^{k-1} p_{\mu_i}(\mu_i | \mu_{i-1}) \right] \sum_{m=1}^{\mu_k-1} p_{\mu_k}(m | \mu_{k-1}) \right\} + \left[\sum_{m=1}^{\mu_K-1} p_{\mu_K}(m | \mu_{K-1}) \right] \prod_{i=1}^{K-1} p_{\mu_i}(\mu_i | \mu_{i-1}). \quad (\text{A13})$$

Comparing Eq. (A10) with Eq. (A13) and using the second last inequality in Eq. (10), $\sum_{m=1}^{\mu_K-1} p_{\mu_K}(m | \mu_{K-1}) < \tilde{u}_K$, we conclude that

$$F(\boldsymbol{\mu}_{l-1}) < u_1. \quad (\text{A14})$$

Combining Eqs. (A12) and (A14), we see that Eq. (10) implies Eq. (7). This concludes the proof of the proposition.

¹M. Kærn, T. C. Elston, W. J. Blake, and J. J. Collins, *Nat. Rev. Genet.* **6**, 451 (2005).

²J. M. Raser and E. K. O’Shea, *Science* **309**, 2010 (2005).

³C. V. Rao, D. M. Wolf, and A. P. Arkin, *Nature (London)* **420**, 231 (2002).

⁴D. T. Gillespie, *J. Comput. Phys.* **22**, 403 (1976).

⁵D. T. Gillespie, *J. Phys. Chem.* **81**, 2340 (1977).

⁶D. T. Gillespie, *Physica A* **188**, 404 (1992).

⁷M. A. Gibson and J. Bruck, *J. Phys. Chem. A* **104**, 1876 (2000).

⁸Y. Cao, H. Li, and L. R. Petzold, *J. Chem. Phys.* **121**, 4059 (2004).

⁹D. T. Gillespie, *J. Chem. Phys.* **115**, 1716 (2001).

¹⁰D. T. Gillespie and L. R. Petzold, *J. Chem. Phys.* **119**, 8229 (2003).

¹¹T. Tian and K. Burrage, *J. Chem. Phys.* **121**, 10356 (2004).

- ¹² A. Chatterjee, D. G. Vlachos, and M. A. Katsoulakis, *J. Chem. Phys.* **122**, 024112 (2005).
- ¹³ M. F. Pettigrew and H. Resat, *J. Chem. Phys.* **126**, 084101 (2007).
- ¹⁴ Z. Xu and X. Cai, *J. Chem. Phys.* **128**, 154112 (2008).
- ¹⁵ A. Auger, P. Chatelain, and P. Koumoutsakos, *J. Chem. Phys.* **125**, 084103 (2006).
- ¹⁶ X. Cai and Z. Xu, *J. Chem. Phys.* **126**, 074102 (2007).
- ¹⁷ S. V. Amari and R. B. Misra, *IEEE Trans. Reliab.* **46**, 519 (1997).
- ¹⁸ W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C* (Cambridge University Press, Cambridge, 1995).
- ¹⁹ Y. Cao, D. T. Gillespie, and L. R. Petzold, *J. Chem. Phys.* **124**, 044109 (2006).
- ²⁰ C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, 2006).
- ²¹ Y. Cao, D. T. Gillespie, and L. R. Petzold, *J. Chem. Phys.* **123**, 054104 (2005).
- ²² A. M. Kierzek, *Bioinformatics* **18**, 470 (2002).
- ²³ X. Cai, *J. Chem. Phys.* **126**, 124108 (2007).