Chemical Engineering Science 137 (2015) 828–836

Contents lists available at ScienceDirect

Chemical Engineering Science

journal homepage: www.elsevier.com/locate/ces

On speeding up stochastic simulations by parallelization of random number generation

Che-Chi Shu, Vu Tran, Jeremy Binagia, Doraiswami Ramkrishna [*,1]

School of Chemical Engineering, Purdue University, West Lafayette, IN 47907, United States

HIGHLIGHTS

- Improved the abstract presentation by adding text to its significance in elucidating the main concept of the paper.
- Included a Table of mathematical quantities with explanation.
- Included a figure explaining further the concept.

ARTICLE INFO

Article history:
Received 22 January 2015
Received in revised form
21 May 2015
Accepted 29 June 2015
Available online 23 July 2015

Keywords:
Stochastic simulation
Tau-leap
Parallel
Chemical processes

ABSTRACT

This paper adds to the tool kit of stochastic simulations based on a very simple idea. Applicable to both SSA and Tau-leap algorithms, it can notably reduce computational times. Stochastic simulations are based on computing sample paths based on the generation of random numbers with either exactly stipulated distribution functions as in SSA (Gillespie, 1977) or in the method of interval of quiescence (Shah et al., 1977) or distribution functions featuring approximations designed to promote efficiency (as in Tau-leap algorithms (Cao et al., 2006; Tian and Burrage, 2004; Peng et al., 2007; Gillespie, 2001; Ramkrishna et al., 2014) where a leap condition with the parameter epsilon is used). The usual strategy involves sequential computation of a large number of sample paths over a bounded time interval which is covered by a set of discrete time subintervals obtained by random number generation. The strategy here departs from the foregoing by parallelizing the generation of random subintervals for the set of sample paths until all sample paths have been computed for the stated time interval. The advantage of this procedure lies in the fact that the time for initiation of the random number generator has been notably reduced. Many examples are demonstrated from SSA as well as Tau-leap algorithms to establish that the advantage of the approach is much more than conceptual.

1. Introduction

The importance of stochastic simulations has risen considerably in recent times both from their applications to biology and investigation of material behavior in the nano state. The repetitive nature of simulations is responsive to simplifications of various kinds. In this paper, we show that the simple strategy of parallelizing random number generations of time subintervals among sample paths can produce notable reductions in computation time.

The idea of the methodology can be communicated in very simple terms although to make a quantitative estimate of the extent of improvement would require an inconvenient amount of effort. Suppose we are interested in computing the behavior of a stochastic process system over a specified time interval. The usual methodology involves exploiting knowledge of the random behavior of the system over successive discrete subintervals by generating random numbers which conform to calculated distributions thus generating a sample path of the process. When many such sample paths are created one after the other, average behavior of the stochastic system as well as fluctuations about the average can be calculated after a suitable number of sample paths have been obtained. The total computational time is clearly governed by the efficiency with which sample paths are created. In what follows, we provide first a simple analysis of the idea to show why the approach is attractive and then demonstrate computational improvements quantitatively with several examples. In showing that a computational procedure has the advantage of being more efficient than an existing one, it is essential to show that for a given computational time the new procedure produces a distinctly more accurate solution.

* Correspondence to: School of Chemical Engineering Purdue University West Lafayette, IN 47907-1283. Tel.: +1 765 494 4066; fax: +1 765 494 0805.
E-mail address: ramkrish@ecn.purdue.edu (D. Ramkrishna).

[1] Harry Creighton Peffer Distinguished Professor.

http://dx.doi.org/10.1016/j.ces.2015.06.066
0009-2509/© 2015 Elsevier Ltd. All rights reserved.

Alternatively, a solution of a specified accuracy must be shown to accrue by the new method with a considerably lighter computational burden. While the foregoing demonstration would certainly be essential to qualify the new procedure, a clearer understanding can be had of the desired comparison by restricting considerations to a simple example, in which it is possible to analytically show why the proposed method is superior. To enable an analytic comparison, we select a simple Poisson process whose properties are well established. In the parallel strategy, will have initiated n sample paths of the process at the outset and allowed to progress simultaneously in time steps. Some paths will progress faster than others. An average time of evolution may be defined (as in Eq. (1) below) to track their concerted motion in time. Those that have transcended the stipulated time will have "dropped off" from the set of n paths. A calculation of the left-over sample paths becomes possible for the Poisson process as also the fluctuations about it. Relating the computation time to the number of steps in the parallel and the sequential strategies, a comparison is enabled. What follows is the translation of this idea in mathematical terms, from which the efficacy of the parallel strategy is elucidated.

2. Analysis

Consider a stochastic process $X(t)$ whose behavior is sought in the interval $I \equiv (0, t_f)$. Given the propensities (transition rates) associated with change, it is possible to define strategies for the simulation of discrete subintervals of time $I_j \equiv (t_{j-1}, t_j)$; $j = 1, 2, \ldots, N$ for the process of interest with the property

$$\bigcup_{j=1}^{N-1} I_j \subset I, \quad \bigcup_{j=1}^{N} I_j \supseteq I, \quad N = 1, 2, \ldots$$

Clearly N is a random integer with an associated probability distribution, say $\pi_k \equiv \Pr\{N = k\}$. The computation of this probability may be tedious but it is not necessary for the ensuing argument. The sample path generated is given by $X(t) = X_j;\ t \in I_j,\ j = 1, 2, \ldots, N$. We presume that n sample paths may be sufficient to obtain reasonable averaging. Further, random number generation is assumed to take *unit* computation time per interval. Thus the computation time taken for the above sample path is N, which results from neglecting possible changes in the computation time for different subintervals. Suppose n sample paths are generated by the usual sequential strategy with the ith sample path involving N_i subintervals of time. Then the computation time for the ith sample path is N_i. The total computation time for the sequential strategy is $\sum_{i=1}^{n} N_i$.

In the parallel strategy we simulate n times, each time subinterval, to produce a *fragment* of the averaged sample path, the computation time for which is n. As we continue this parallel strategy of computing the entire collection of sample paths in fragments, automatically ending those paths that have reached or exceeded the targeted time interval, the number of random variables to be generated can be seen to diminish progressively with a corresponding decrease in computation time.

Suppose for the sake of a preliminary demonstration, we restrict ourselves to processes for which the time interval for the next change at any stage is independent of its current state, (e.g., the Poisson process). We will be concerned with a collection of n sample paths to evolve by simulation in discrete stages over time. In other words, each simulation will lead to a jump in the process over a time subinterval. Those sample paths that reach or transcend time t_f will have been completed and excluded from further evolution. Thus we begin with a *population* of n sample paths at the outset of the simulation. At the end of the first simulation step, the n different sample paths will have evolved to different times, say $\tau_1, \tau_2, \ldots, \tau_n$. Since this first simulation step is to be followed by numerous additional ones, the evolution of each sample path after the ith step may be described by $\left\{\tau_1^i, \tau_2^i, \ldots, \tau_{K_i}^i\right\}$, where K_i is the random number of sample paths left in the collection after $(n - K_i)$ sample paths have transcended the interval $(0, t_f)$ during the i simulation steps. Since stochastic simulations of this type are contingent on the existence of a cumulative distribution function $F_T(\tau)$ for the time at which a change occurs, the jth sample path in this collection which has evolved to time τ_j^i before arriving at the ith will have an exit probability of $\mu_j^i \equiv \int_{t_f - \tau_j^i}^{\infty} dF_T(\tau)$.

We seek to identify an equation for the *population* of sample paths in the collection with reference to some *average* time of evolution (rather than their individual times of evolution) to represent *all* the sample paths in a given step. Towards this end, a probability would be required for any sample path in the collection to quit by transcending the interval $[0, t_f]$. We adopt the average time for the ith step denoted $\bar{\tau}_i$ as below which will hold for the sample paths which survive for the next step.

$$\bar{\tau}_i \equiv \bar{\tau}_{i-1} + \int_0^{t_f - \bar{\tau}_{i-1}} \tau dF_T(\tau) \Big/ \int_0^{t_f - \bar{\tau}_{i-1}} dF_T(\tau), \quad \bar{\tau}_0 = 0 \tag{1}$$

The foregoing choice represents the average time at which sample paths in the ith step have expended an average time of $\bar{\tau}_{i-1}$ in the previous step. The probability μ_i that a sample path exits in the ith step from the collection may now be estimated as

$$\mu_i = \int_{t_f - \bar{\tau}_i}^{\infty} dF_T(\tau) \tag{2}$$

If we now let $P_k^i \equiv \Pr\{K_i = k | K_1 = n\}$, then it is readily shown that

$$P_k^{i+1} = (1 - \mu_i)^k \sum_{r=0}^{n-k} \binom{k+r}{r} \mu_i^r P_{k+r}^i, \quad P_k^0 = \delta_{k,n} \tag{3}$$

Eq. (3) uses the initial condition that there are n sample paths in all for which the subintervals are generated and may be rewritten as

$$P_k^{i+1} = (1 - \mu_i)^k \sum_{m=k}^{n} \binom{m}{k} \mu_i^{m-k} P_m^i \tag{4}$$

The above equation is solved in the Supplementary material S.1 to obtain the first and second moments of K_{i+1}.

$$EK_{i+1} = = n \prod_{j=0}^{i} (1 - \mu_j) \tag{5}$$

$$EK_{i+1}^2 = n \prod_{j=0}^{i} (1 - \mu_j)^2 \left[n + \sum_{k=0}^{i} \mu_k \prod_{j=0}^{k} (1 - \mu_j)^{-1} \right] \tag{6}$$

The variance of K_{i+1}, denoted $V(K_{i+1})$, is obtained from (5) and (6)

$$V(K_{i+1}) = EK_{i+1}^2 - (EK_{i+1})^2 = n \prod_{j=0}^{i} (1 - \mu_j)^2 \left[\sum_{k=0}^{i} \mu_k \prod_{j=0}^{k} (1 - \mu_j)^{-1} \right] \tag{7}$$

from which the coefficient of variation, denoted COV_i, is obtained as

$$COV_i = \frac{\sqrt{V(K_{i+1})}}{EK_{i+1}} = \sqrt{n^{-1} \sum_{k=0}^{i} \mu_k \prod_{j=0}^{k} (1 - \mu_j)^{-1}} \tag{8}$$

The computation time for the parallel strategy may now be estimated from $\sum_{i=0}^{i_{\max}} EK_{i+1}$, where $i_{\max}$ may be chosen by requiring that the above $EK_{i_{\max}+1}$ is suitably small, which implies that n sample paths have been cleared from the collection. The coefficient of variation (8) serves to verify that the fluctuation associated with this population is negligible.

We now consider the very simple case of the Poisson process adding to a population of individuals at mean rate λ; the change in this process over time is the addition of an individual (in this case independently of the prior population) which has the distribution function given by $F_T(\tau) = 1 - e^{-\lambda\tau}$. For this case, we have from (2)

$$\mu_i = \lambda \int_{t_f - \overline{\tau}_i}^{\infty} e^{-\lambda\tau}\, d\tau = e^{-\lambda(t_f - \overline{\tau}_i)} \tag{9}$$

The expression (9) shows that, as $\overline{\tau}_i$ approaches t_f, μ_i approaches 1 making the sample path increasingly likely to exit the collection. From (1), we obtain the following for the Poisson process.

$$\overline{\tau}_i = \overline{\tau}_{i-1} + \frac{1}{\lambda} - \frac{e^{-\lambda(t_f - \overline{\tau}_{i-1})}\left(t_f - \overline{\tau}_{i-1}\right)}{1 - e^{-\lambda(t_f - \overline{\tau}_{i-1})}}, \quad \overline{\tau}_0 = 0 \tag{10}$$

For a preliminary quantitative demonstration, we consider simulating the Poisson process by the sequential as well as the parallel strategy. We have $F_T(\tau) = 1 - e^{-\lambda\tau}$ to generate subintervals.

For the sequential strategy using the Poisson process, it is possible to readily obtain the probability distribution π for the discrete random variable N (see Supplementary material S.2), the number of subintervals which sum to cover the time interval $[0, t_f]$. Thus

$$\pi_k \equiv \Pr\{Sample\ path\ ends | N = k\} = 1 - \frac{1}{2^k}\left(1 - e^{-\lambda t_f}\right)^{k+1}, k = 0, 1, 2, \ldots$$

For the parallel strategy, we generate random numbers satisfying the cumulative distribution function $F(\tau) = 1 - e^{-\lambda\tau}$, through simulation of the uniform random variable $X =^{d} U[0, 1]$.

2.1. Comparison of computation times: sequential and parallel strategy

The expected computation time for the sequential strategy has been shown to be given by

$$E\sum_{k=1}^{n} N_k = \sum_{i=1}^{n} EN_k = nEN \tag{11}$$

which follows from the sample paths possessing the *same* distribution function for generation of subintervals. EN, being the expected number of subintervals in generating an entire sample path, is given in the Supplementary material (S.3.2) reproduced below.

$$EN = \frac{\frac{1}{2}K_{\min}(K_{\min}+1) - 2(1-r)^{-2}}{K_{\min} - 2r(1-r)^{-1}}, \quad r \equiv \frac{1}{2}\left(1 - e^{-\lambda t_f}\right) \tag{12}$$

where $K_{\min}$ is as specified by Supplementary material (S.3.3). Thus the expected computation time for the simulation of the Poisson process using the sequential strategy is specified by (12).

Using the parallel strategy, the expected computational time for the Poisson process is given by

$$n\sum_{i=0}^{i_{\max}} \prod_{j=0}^{i} \left(1 - e^{-\lambda(t_f - \overline{\tau}_j)}\right) \tag{13}$$

where $\overline{\tau}_j$ is given by Supplementary material (S.4.3). The ratio of (13) to (11) provides a good quantitative measure of the effectiveness of the parallel strategy relative to the sequential strategy as it is a direct comparison of the expected computation times. Fig. 1 shows the results of calculations. In this figure, simulation time was plotted as a function of square root of $(\lambda * t_f)$, and it clearly indicates that the sequential algorithm would cost more CPU time than the parallel algorithm at any given value of $(\lambda\, t_f)$.

The demonstration above was made for processes in which the distribution function for successive generation of quiescence intervals was the same. For many applications, this is not a realistic assumption so that a demonstration of the effectiveness of the parallel strategy necessarily requires detailed simulations by both strategies. Fig. 2 below provides a more detailed schematic picture of how each method works.

Let us discuss strategies utilized in each method for a simple scenario of simulations that is composed of 4 sample paths, shown in the figure above. The sequential method simulates leaps sequentially and keeps updating new states using information from the previous step. This procedure is iterated until reaching the final t_f. Upon the completion of one, it then can be applied to the next sample path. The parallel method, on the other hand, will start with generating the first leap for each trajectory independently. Second leap for each sample will then be simulated simultaneously and applied to update variables

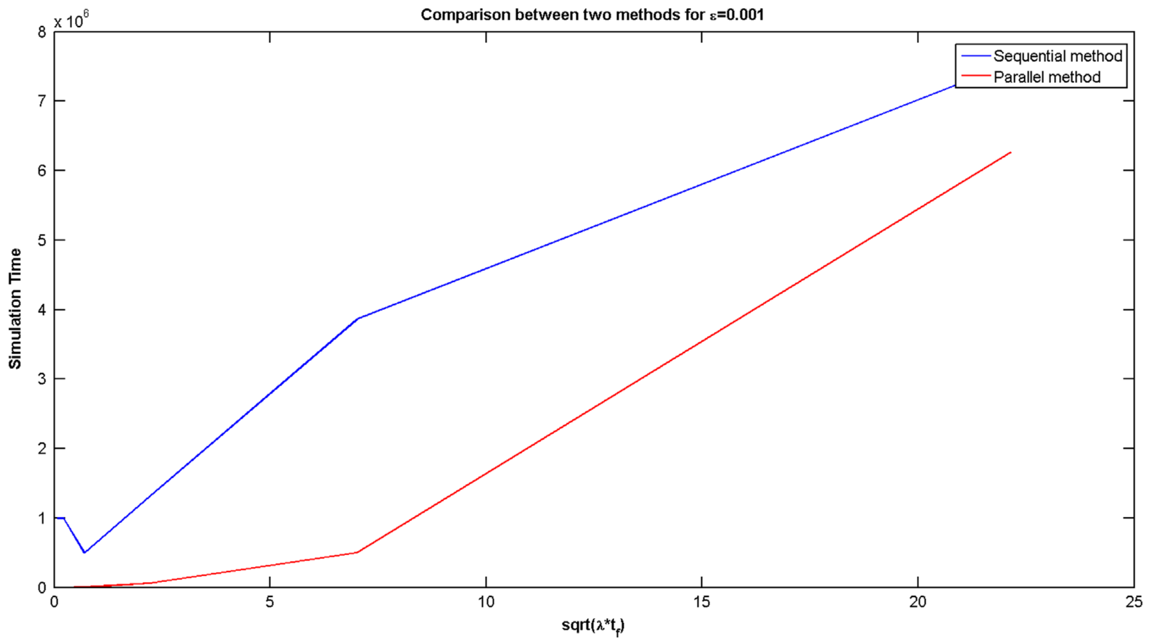

Fig. 1. Comparison between two methods for eps=0.001.

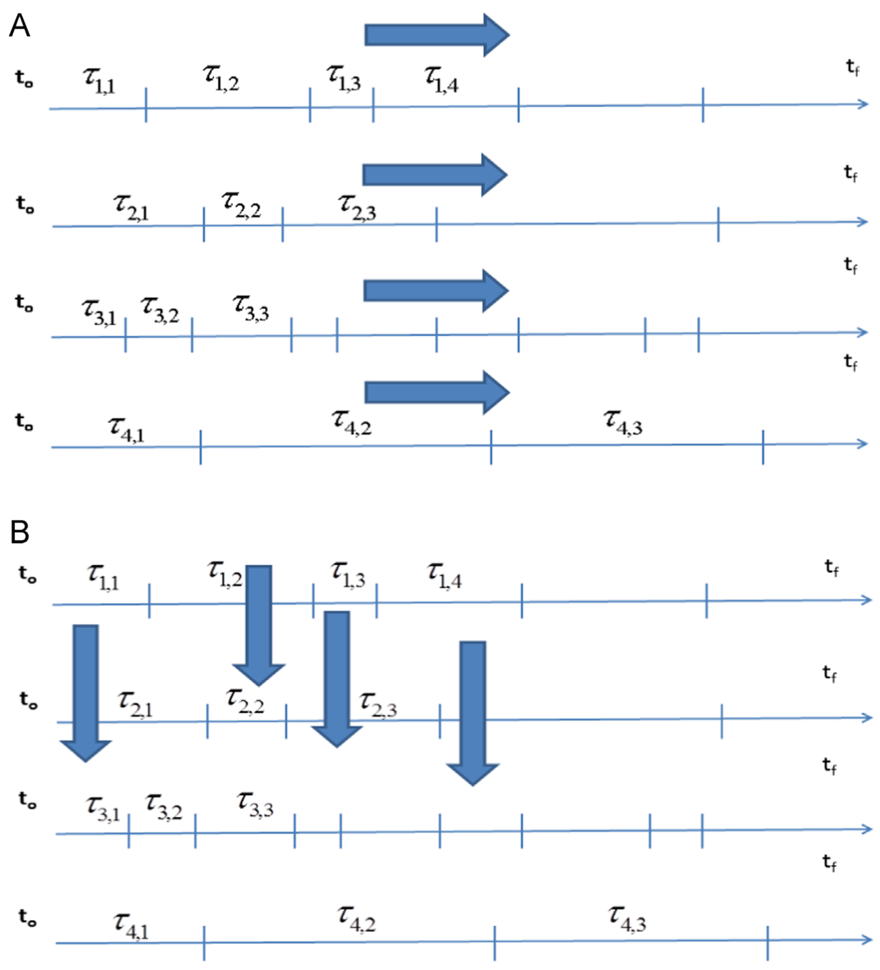

Fig. 2. Comparison of (A) Sequential method and (B) Parallel method.

that correspond to the previous states from the same sample path. This procedure is carried on iteratively. Since generation of various sample paths is independent, some sample paths will reach the mature time before others. Due to that nature, the parallel method can reduce the number of trajectories that need to be simulated as it approaches the final time. Specifically, in Fig. 2B, it clearly indicates that the sample path 4 can be dropped out of the simulation bath after 4 steps, followed by sample path 1 after another 2 steps. The number sample path will keep decreasing as the simulation evolves with time, hence reducing memory burden and CPU time. The strategy can also be presented in a step-wise manner in the Section 3. To further illustrate the key idea, in Sections 4 and 5, simulation results corresponding to several examples are shown and discussed.

2.2. Parallelizing tau-leaping algorithm

- For a system with n sample paths to evolve, at time zero, set the state of time for each sample path $t_1, t_2, ..., t_n$ equal to 0.
- Set up the initial numbers of molecules for each sample path, say $X_1(t_1=0)=x_1^o$; $X_2(t_2=0)=x_2^o$; ...; $X_n(t_n=0)=x_n^o$ where the superscript o denotes the initial value. Note that X_k, $k=1,2...,n$ is a vector describing the states of the system for the sample path k.
- Calculate the tau values $\tau_1, \tau_2, ..., \tau_n$ for each sample path and accordingly generate poison random number for all reactions of all sample paths. With information of stoichiometry, the change of states for each sample path could be obtained as δx_1; $\delta x_2, ..., \delta x_n$. Note that δx_k, $k=1,2...,n$ is a vector describing the state change of the system for the sample path k.
- Update the states for each sample path by $X_1(t_1+\tau_1)=X_1(t_1)+\delta x_1$; $X_2(t_2+\tau_2)=X_2(t_2)+\delta x_2$; ...; $X_n(t_n)=\delta x_n$
- Update the state of time for each sample path by $t_1 \leftarrow t_1+\tau_1$, $t_2 \leftarrow t_2+\tau_2, ...,$ $t_n \leftarrow t_n+\tau_n$
- For sample path k with $t_k < t_f$, repeat step 3 to 5.
- The calculation is done when $t_k \geq t_f$ for all $k \in [1,2...,n]$.

3. Examples

Example 1. Schlogl's chemical reaction model (Cao et al., 2006; Gillespie, 2001; Ramkrishna et al., 2014), known for its bistability.B_1 and B_2 are constants with their particle numbers as N_1 and N_2 respectively.

$$B_1+2X \underset{c_2}{\overset{c_1}{\rightleftharpoons}} 3X, \quad B_2 \underset{c_4}{\overset{c_3}{\rightleftharpoons}} X \tag{14}$$

The values of parameters, adapted from Cao et al.'s paper and Gillespie's paper (Cao et al., 2006; Gillespie 1977), $c_1=3\times10^{-7}$, $c_2=10^{-4}$, $c_3=10^{-3}$, $c_4=3.5$, $N_1=1\times10^5$, and $N_2=2\times10^5$. The initial condition of X is 250 and we simulate the system up to $t_f=4$.

Example 2. Consecutive linear reaction system (Cao et al., 2006; Tian and Burrage, 2004; Peng et al., 2007)

$$X_1 \overset{c_1}{\Longrightarrow} X_2 \overset{c_2}{\Longrightarrow} X_3 \tag{15}$$

Rate constants $c_1=1$ and $c_2=1$, and initial conditions $X_1=10^4$, $X_2=1$ and $X_3=0$, we calculate X_3 until time is equal to 0.1.

Example 3. Dimer (Cai and Xu, 2007) $S_1 \overset{c_1}{\Longrightarrow} \phi, \quad S_2 + S_3 \overset{c_2}{\Longrightarrow} S_4$

At $t = 0$, $S_1 = 3000$, $S_2 = 3000$, $S_3 = 10^4$. Moreover, $c_1 = 1$ and $c_2 = 10^{-4}$. Our goal is to compare the results between distributions generated by two algorithms at $t_f = 2$.

Example 4. In this system, a monomer S_1 can dimerize to an unstable form S_2, which in turn coverts to a stable form S_3. Since S_2 is unstable, it can also convert back to S_1 (Cao et al., 2006; Gillespie and Petzold, 2003).

$$S_1 \overset{c_1}{\Longrightarrow} \phi, \quad S_1 + S_1 \underset{c_3}{\overset{c_2}{\rightleftharpoons}} S_2, \quad S_2 \overset{c_4}{\Longrightarrow} S_3$$

Initial conditions: $S_1(0) = 4150$, $S_2(0) = 39564$, $S_3(0) = 3445$
Rate constants: $c_1 = 1$, $c_2 = 0.002$, $c_3 = 0.5$, $c_4 = 0.04$
Our goal is to collect the distribution of the dimer at $t_f = 10$.

4. Results and discussion

Four examples have been utilized to compare the effectiveness of the proposed parallelization, also referred to here as the simultaneous algorithm. The first example was that of Schologl's system, for which comparison was made of simulations with the τ-leap method involving Poisson distribution. Fig. 3, shows consistent results for the distribution of X, by both methods, as the two curves virtually overlap one another over the entire range. In Figs. 4 and 5, performances of the two algorithms are compared in terms of CPU time. Clearly, the sequential method requires substantially longer computation times for the simulation, than the simultaneous algorithm. For instance, with 30,000 trajectories, the sequential algorithm ran about 50 times slower than the other. In example 2, the binomial τ-leap method was used for comparison, and a similar trend is seen in Figs. 6 and 7. The simultaneous algorithm outperforms the sequential with a 120 fold improvement in CPU time. Figs. 8–10 were produced for example 3. Fig. 8 compared the accuracy of each solution generated by the two algorithms to that produced by SSA with 50,000 trajectories. To fully investigate the benefit of this method, the performances were compared from two different aspects: in Fig. 9 epsilon, which represents the measure of accuracy in the tau-leap algorithm (Cao et al., 2006; Peng et al., 2007; Gillespie, 2001), was fixed and the number of trajectories was changed and in Fig. 10 the number of trajectories was fixed and epsilon was varied. In either case, the

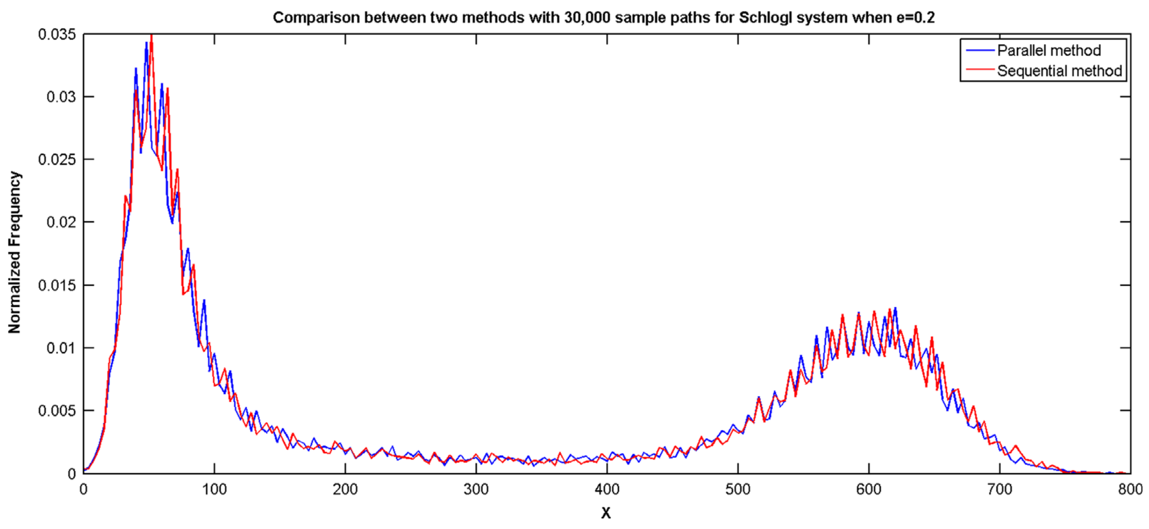

Fig. 3. Comparison between two methods with 30,000 sample paths for Schogl system when $e = 0.2$.

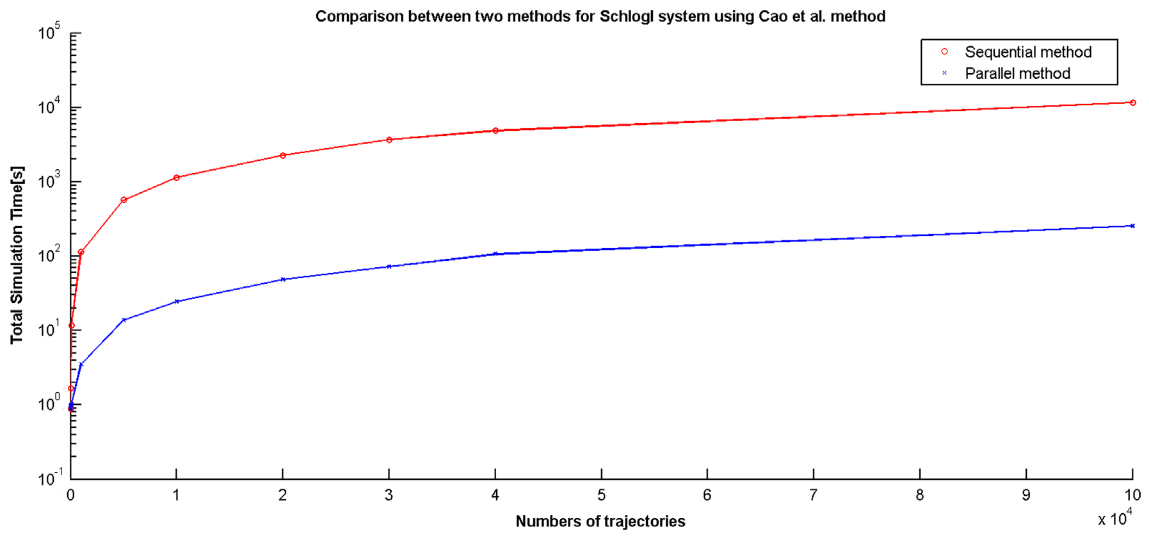

Fig. 4. Comparison between two methods for Schlogl system using Cao et al. method.

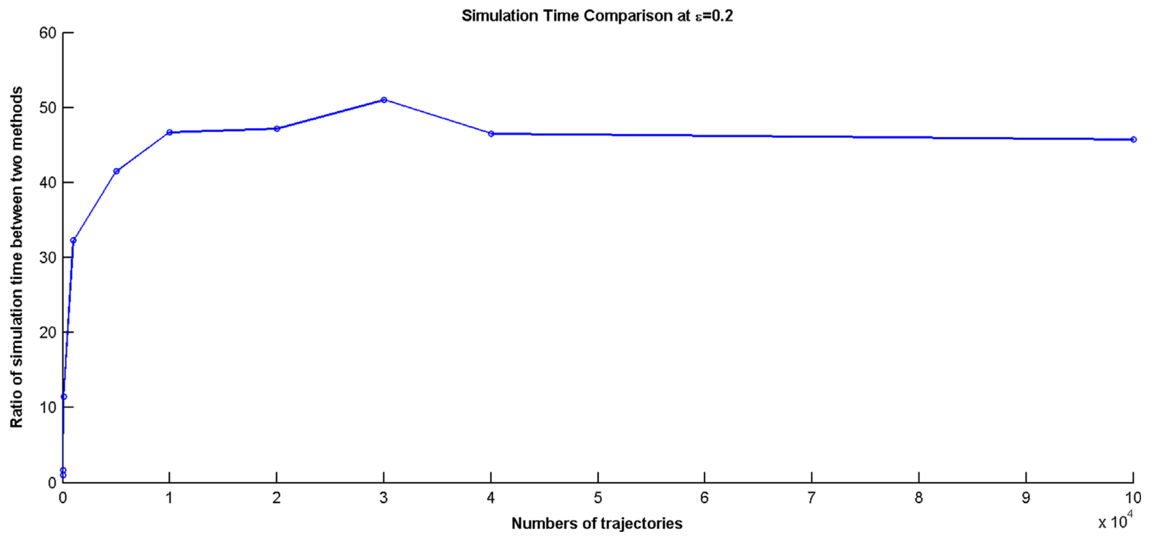

Fig. 5. Simulation time comparison at $e = 0.2$.

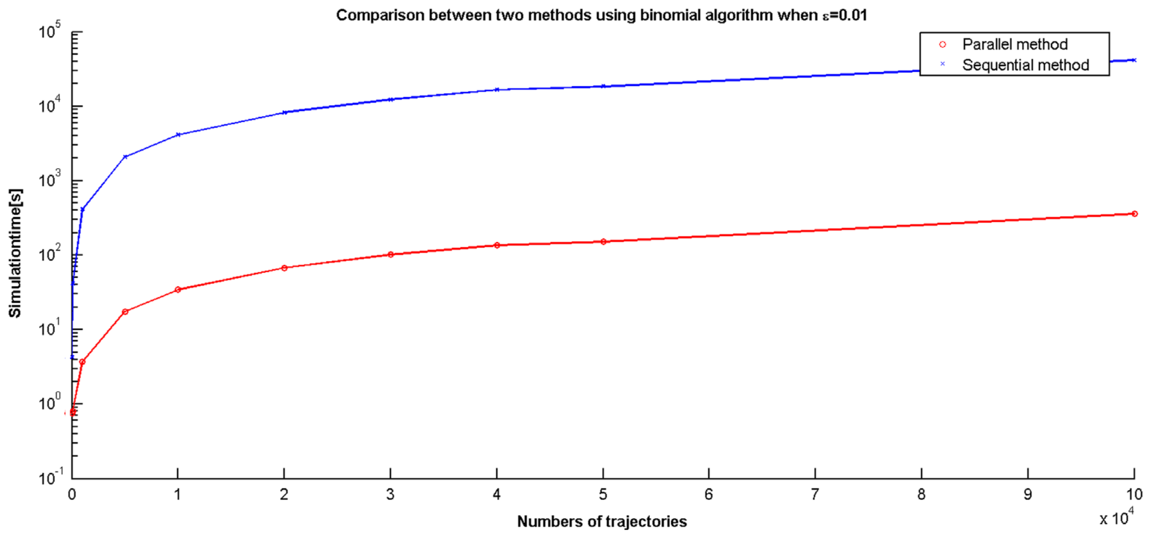

Fig. 6. Comparison between two methods using binomial algorithm when e=0.01.

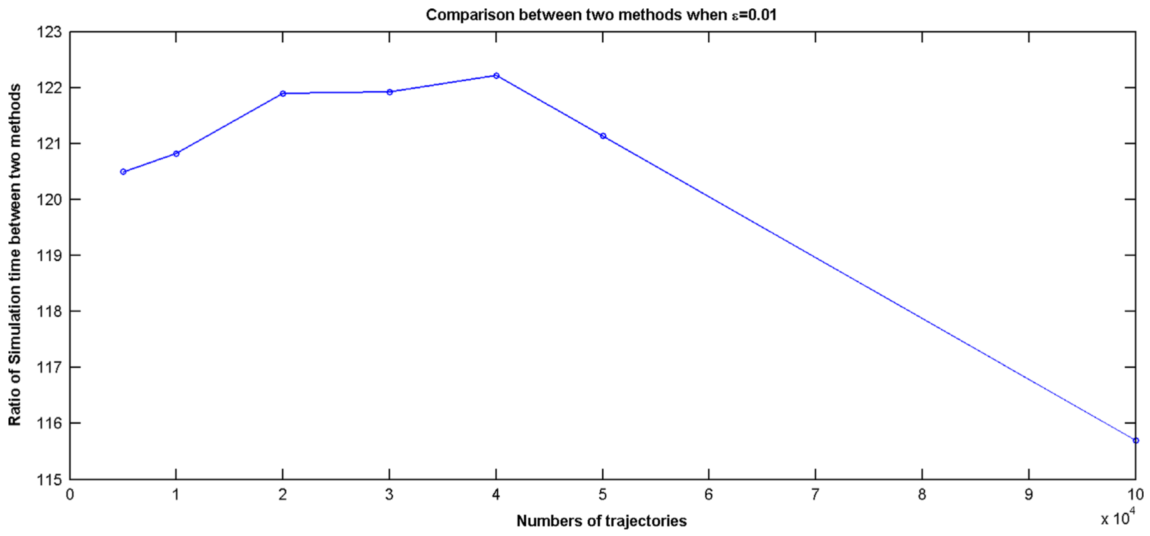

Fig. 7. Comparison between two methods when e=0.01.

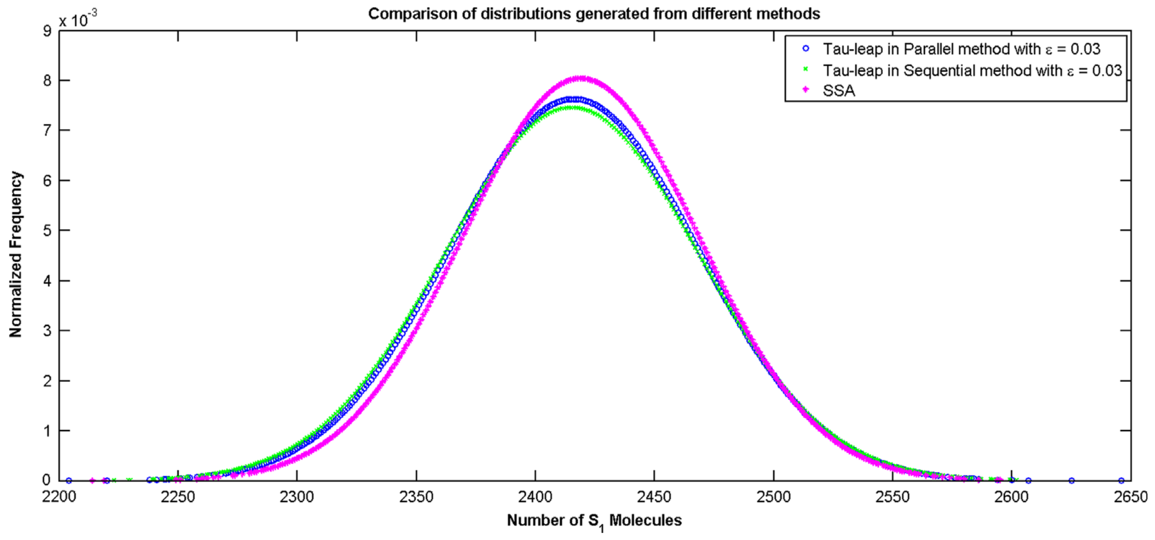

Fig. 8. Comparison of distribution generated from different methods.

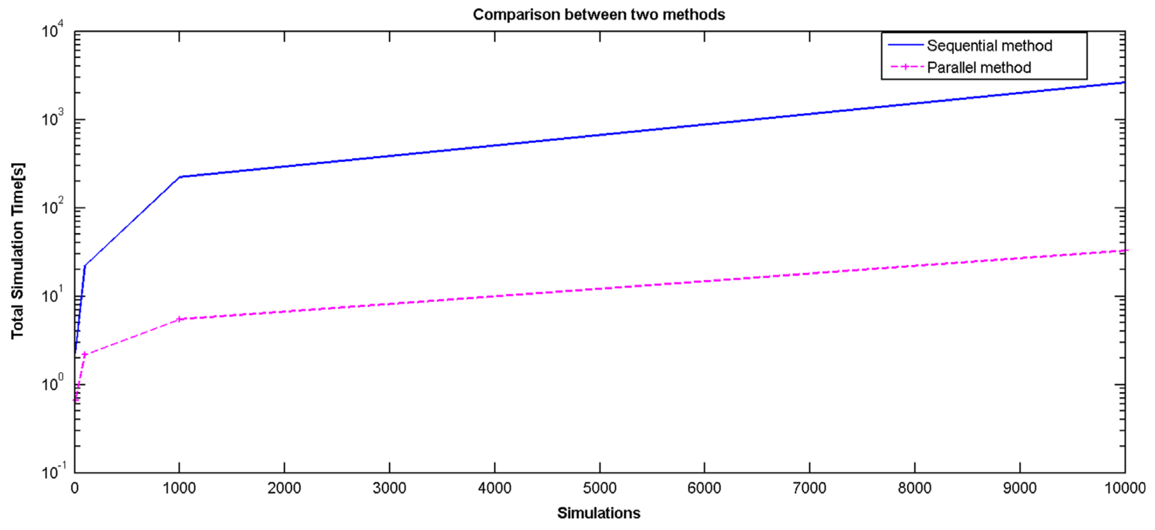

Fig. 9. Comparison between two methods.

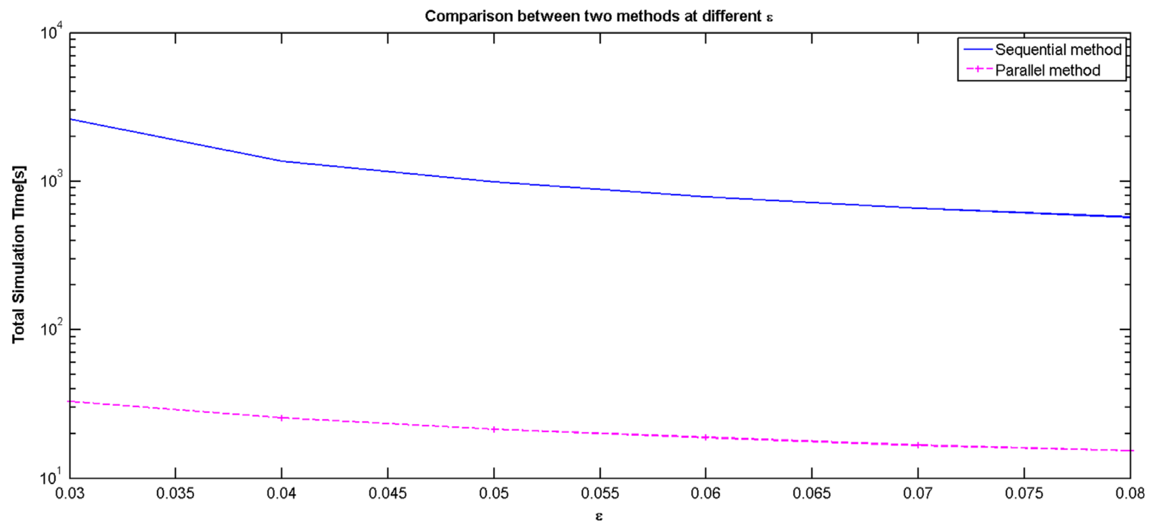

Fig. 10. Comparison between two methods at different epsilon.

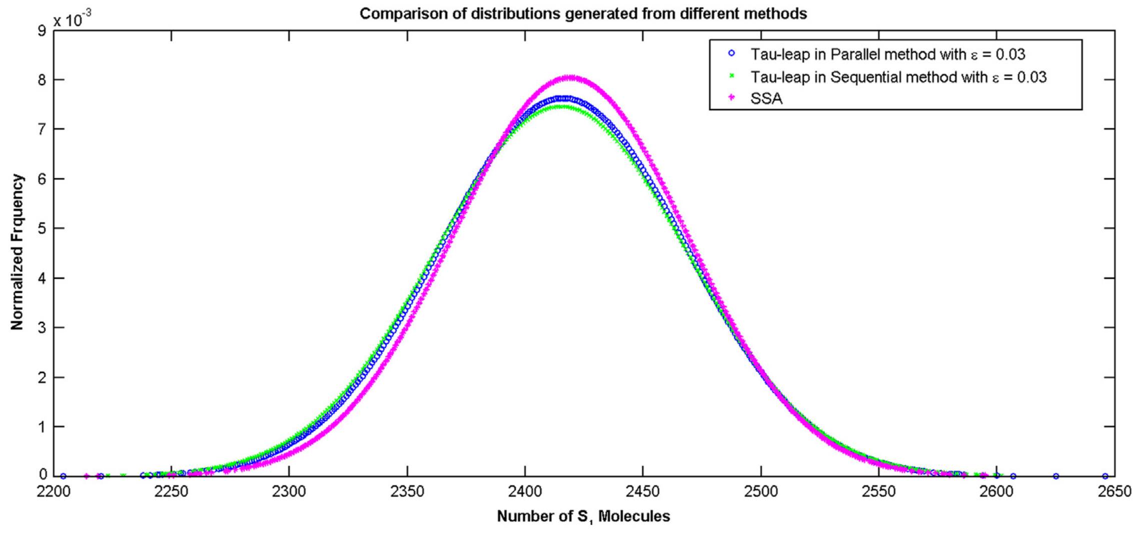

Fig. 11. Comparison of distribution generated from different methods.

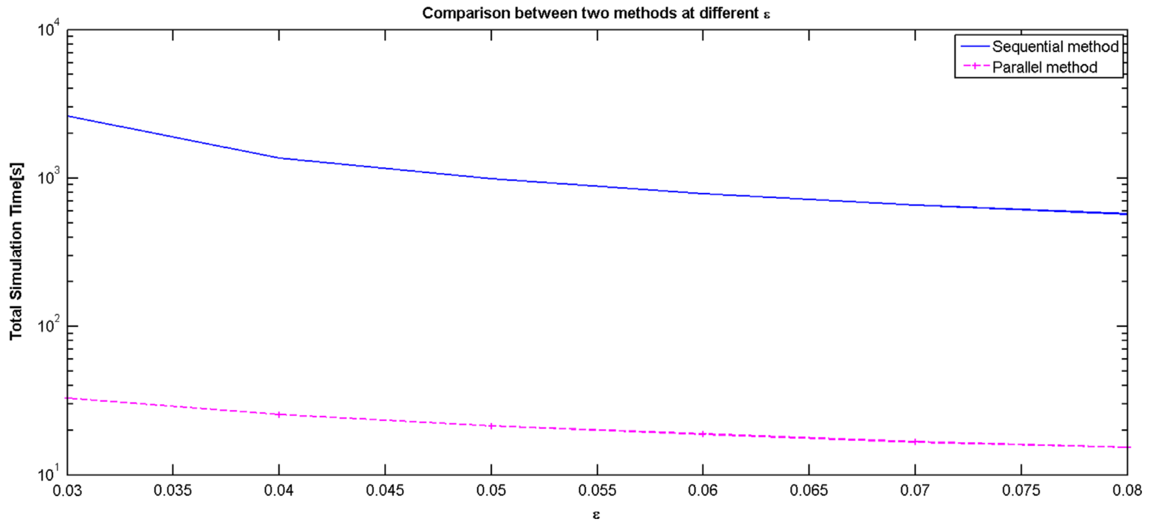

Fig. 12. Comparison between two methods at different epsilon.

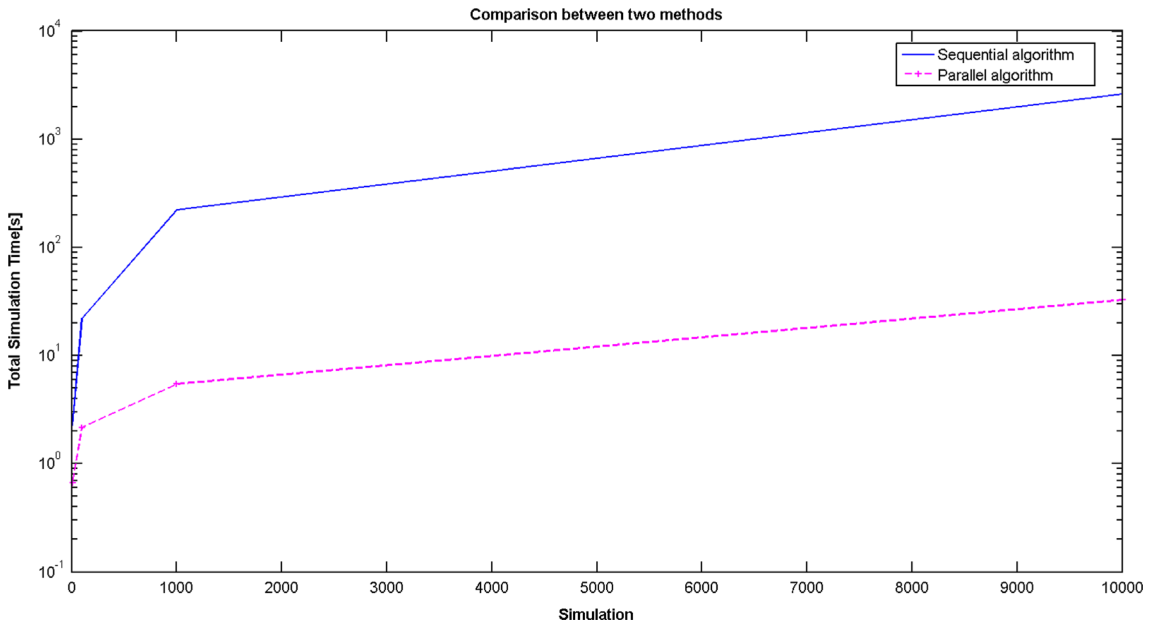

Fig. 13. Comparison between two methods.

simultaneous method has prevailed unambiguously over the other. Figs. 11–13, display simulations for example 4, also show notably less CPU time to produce the same results.

5. Conclusions

This paper establishes that the parallel strategy for generating sample paths of stochastic processes is notably superior to the usual sequential strategy followed in stochastic simulations. Since each trajectory is independent of all the others and can vary greatly in the number of steps within each sample path, simultaneous generation of several such leaps for different trajectories eliminates the delay time between each trajectories. As a result, for a given requirement of accuracy, the overall simulation time can be optimized.

Symbol description:

Symbol	Description
t_f	Predefined final time
τ_j^i	Evolution of sample path j after i steps
μ_j^i	Exit probability after i steps in the j-th sample path
λ	Mean rate

Acknowledgments

This work was supported by National Institutes of Health Grants GM081888 (to W.-S.H.) and SURF grant to Jeremy Binagia

Appendix A. Supporting information

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.ces.2015.06.066.

References

Cai, X., Xu, Z., 2007. K-leap method for accelerating stochastic simulation of coupled chemical reaction. J. Phys. Chem. 126, 074102.
Cao, Y., Gillespie, D.T., Petzold, L.R., 2006. Efficient step size selection for the tau-leaping simulation method. J. Phys. Chem. 124, 4 [accessed 04.09.14].
Gillespie, D.T., 1977. Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. 81, 25 [accessed 04.09.14].
Gillespie, D.T., 2001. Approximate accelerated stochastic simulation of chemically reacting systems. J. Phys. Chem. 115, 1716 [accessed 04.09.14].
Gillespie, D.T., Petzold, L.R., 2003. Improved leap-size selection for accelerated stochastic simulation. J. Phys. Chem. 119, 8229–8234.
Peng, X., Zhou, W., Wang, Y., 2007. Efficient binomial leap method for simulating chemical kinetics. J. Phys. Chem. 126, 224109 [accessed 04.09.14].
Ramkrishna, D., Shu, C.C, Tran, V., 2014. New tau-leap strategy for accelerated stochastic simulation. Ind. Eng. Chem. Res. 53 (49), 18975–18981.
Shah, B.H., Ramkrishna, D., Borwanker, J.D., 1977. Simulation of particulate systems using the concept of the interval of quiescence. AIChe J 23, 897–904 [accessed 04.09.14].
Tian, T., Burrage, K., 2004. Binomial leap methods for simulating stochastic chemical kinetics. J. Phys. Chem. 121, 10356–10364 [accessed 04.09.14].