Cell Cycle Modeling for Budding Yeast with Stochastic Simulation Algorithms

Tae-Hyuk Ahn*[1], Layne T Watson[1,2], Yang Cao[1], Clifford A Shaffer[1], and William T Baumann[3]

[1]Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA
[2]Department of Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA
[3]Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA

Email: Tae-Hyuk Ahn* – thahn@cs.vt.edu

*Corresponding author

Abstract

Background: For biochemical systems, where some chemical species are represented by small numbers of molecules, discrete and stochastic approaches are more appropriate than continuous and deterministic approaches. The continuous deterministic approach using ordinary differential equations is adequate for understanding the average behavior of cells, while the discrete stochastic approach accurately captures noisy events in the growth-division cycle. Since the emergence of the stochastic simulation algorithm (SSA) by Gillespie, alternative algorithms have been developed whose goal is to improve the computational efficiency of the SSA.

Results: This paper explains and empirically compares the performance of some of these SSA alternatives on a realistic model. The budding yeast cell cycle provides an excellent example of the need for modeling stochastic effects in mathematical modeling of biochemical reactions. This paper presents a stochastic approximation of the cell cycle for budding yeast using Gillespie's stochastic simulation algorithm. To compare the stochastic results with the average behavior, the simulation must be run thousands of times.

Conclusions: Many of the proposed techniques to accelerate the SSA are not effective on the budding yeast problem, because of the scale of the problem or because underlying assumptions are not satisfied. A load balancing algorithm improved overall performance on a parallel supercomputer.

Background

The cell-division cycle is the sequence of events that take place in a eukaryotic cell leading to its replication. A growing cell replicates all its components and divides them into two daughter cells, so that each daughter has the information and machinery necessary to repeat the process [1]. Mathematical modeling and computational methods are needed to understand complex yeast control

systems. Deterministic mathematical modeling for the budding yeast cell cycle gives the average behavior of populations of dividing cells [2]. However, some major regulatory proteins occur in small numbers such that minor changes in timing and reaction rates can have major inputs on outcomes. Thus, the stochastic approach provides more accurate results than does the deterministic one. In addition, when cell cycle controls are compromised by mutation, random fluctuations are important for modeling the effects of the mutants. Therefore, it is desirable to translate a deterministic cell cycle model into a stochastic model, and simulate the model with an appropriate stochastic method.

Gillespie's stochastic simulation algorithm (SSA) ([3], [4]) is a well-known algorithm using Monte Carlo methods to simulate the chemical reactions. The SSA is an asymptotically exact stochastic method to simulate chemical systems, but the SSA is often slow because it simulates every reaction. Since the SSA emerged, there have been many attempts to improve the computational efficiency ([5], [6], [7]), however, the core principles remain the same.

One notable attempt to improve the SSA is the tau-leaping method [6]. Tau-leaping attempts to achieve increased computational efficiency by leaping over many fast reactions. The implicit tau-leaping method compensates for difficulty with stiff systems [8]. Stiff systems are characterized by well separated fast and slow time scales in a dynamic system, the fastest of which is stable. Some approaches try to reduce time consumption with different assumptions such as quasi steady-state approximation (QSSA) [9] and total quasi steady-state approximation (tQSSA) [10] for stiff systems. This paper compares computational efficiency and exactness between SSA, tau-leaping, implicit tau-leaping, QSSA, and tQSSA based on numerical experiments first with a model using simple chemical reactions and stiff systems, and then with the budding yeast model. StochKit [11] is used to do stochastic simulation of the budding yeast model. Because StochKit supports various approximate simulation methods based on the SSA such as explicit and implicit tau-leaping methods, the computational efficiency of the approximation methods can be compared easily by using StochKit.

Stochastic methods require that the model be cast in terms of population because they consider reactions with individual molecules. The problem is, however, that ODE models are usually based on concentration values. Therefore, the concentration-based model has to be changed into a population-based model to simulate using a stochastic method. Previous work [12] explained the conversion process using JigCell [13] in detail. StochKit [11] is used to do stochastic simulation of the converted budding yeast model. StochKit supports various approximate simulation methods based on the SSA, but only the exact SSA is used to get precise results.

Because the SSA simulates every time step, the SSA is much slower than a deterministic simulation. Moreover, the simulation must be run thousands of times to generate enough data to determine the correct distribution of the behavior. Therefore, it is desirable to run many independent SSA simulations in parallel. StochKit supports MPI for parallel SSA runs, but the user assigns jobs for each processor. Sometimes, the

processor times for individual runs are quite different. The result is poor parallel efficiency. This paper presents a dynamic load balancing algorithm that improves the parallel efficiency.

The SSA and approximation methods are explained in the next section. The budding yeast cell cycle model and the dynamic load balancing algorithm are presented next. Finally new biological results and numerical comparisons are given.

Stochastic Simulation Algorithms

SSA

Suppose a biochemical system or pathway involves N molecular species $\{S_1, ..., S_N\}$. $X_i(t)$ denotes the number of molecules of species S_i at time t. People would like to study the evolution of the state vector $X(t) = (X_1(t), ..., X_N(t))$ given that the system was initially in the state vector $X(t_0)$. Suppose the system is composed of M reaction channels $\{R_1, ..., R_M\}$. In a constant volume Ω, assume that the system is well-stirred and in thermal equilibrium at some constant temperature. There are two important quantities in reaction channels R_j: the state change vector $v_j = (v_{1j}, ..., v_{Nj})$, and propensity function a_j. v_{ij} is defined as the change in the S_i molecules' population caused by one R_j reaction, and $a_j(x)dt$ gives the probability that one R_j reaction will occur in the next infinitesimal time interval $[t, t + dt)$.

The SSA simulates every reaction event ([3], [4]). With $X(t) = x$, $p(\tau, j|x, t)d\tau$ is defined as the probability that the next reaction in the system will occur in the infinitesimal time interval $[t+\tau, t+\tau+d\tau)$, and will be an R_j reaction. By letting $a_0(x) \equiv \sum_{j=1}^{M} a_j(x)$, the equation

$$p(\tau, j|x, t) = a_j(x) \exp(-a_0(x)\tau),$$

can be obtained. On each step of the SSA, two random numbers r_1 and r_2 are generated from the uniform (0,1) distribution. From probability theory, the time for the next reaction to occur is given by $t + \tau$, where

$$\tau = \frac{1}{a_0(\mathbf{x})} \ln(\frac{1}{r_1}).$$

The next reaction index j is given by the smallest integer satisfying

$$\sum_{j'=1}^{j} a_{j'}(x) > r_2 a_0(x).$$

After τ and j are obtained, the system states are updated by $X(t + \tau) := x + v_j$, and the time is updated by $t := t + \tau$. This simulation iteration proceeds until the time t reaches the final time.

Explicit tau-leaping

The SSA is an exact stochastic method for chemical reactions, however, it is very slow for many practical systems because the SSA simulates one reaction at a time. One approximate simulation approach is *tau-leaping* [6]. The basic idea of the tau-leaping method is that many reactions can be simulated at each step with a preselected time τ. The tau-leaping method requires that the selected τ must be small enough to satisfy the "leap condition": The expected state change induced by the leap must be sufficiently small that propensity functions remain nearly constant during the time step τ.

$K_j(\tau; x, t)$ is defined as the number of times, given $X(t) = x$, that reaction channel R_j will fire in the time interval $[t, t+\tau)$ where $j = 1, \ldots, M$. If $X(t) = x$, then the state can be updated by

$$X(t+\tau) = x + \sum_{j=1}^{M} K_j(\tau; x, t) v_j.$$

$K_j(\tau; x, t)$ is modeled by a Poisson random variate. The explicit tau-leaping method assumes

$$K_j(\tau; x, t) = P_j(a_j(x)\tau),$$

where P_j is a Poisson random variate with mean and variance $a_j(x)\tau$.

In order to select the largest value of τ that satisfies the leap condition, the Jacobian matrices for the propensity functions are used ([6], [14]). One new approach is to select τ such that relative changes in the propensity functions are bounded [15]. This new τ selection procedure is faster and more accurate than previous methods. Therefore, the explicit tau-leaping method proceeds as follows. Select a τ that satisfies the leap condition. Generate the Poisson random variables for each reaction and adjust the leap time by $t := t + \tau$ and the states by $X(t+\tau) := x + \sum_{j=1}^{M} K_j(\tau; x, t) v_j$. This simulation iteration also proceeds until the time t reaches the final time t_f.

Implicit tau-leaping

Implicit tau-leaping addresses the shortcomings of explicit tau-leaping when the systems are stiff [8]. Stiff systems are characterized by well separated fast and slow time scales in a dynamic system, with the fast mode being stable. In a stiff system, solutions by explicit tau-leaping are unstable unless the time stepsize τ is kept smaller than the smallest (fastest) time scale in the system [8].

Tau-leaping as described previously is an explicit method because the propensity functions a_j are evaluated at the current known state x. Therefore, the future state $X(t+\tau)$ is an explicit function of $X(t)$, and the states can be updated by the equation,

$$X^{et}(t+\tau) = x + \sum_{j=1}^{M} v_j P_j(a_j(x)\tau),$$

where the superscript "et" indicates that the explicit method is used. The implicit method is described by

$$X^{it}(t+\tau) =$$
$$x + \sum_{j=1}^{M} v_j \Big[\tau a_j\big(X^{it}(t+\tau)\big) + P_j(a_j(x)\tau) - \tau a_j(x) \Big],$$

where the superscript "it" stands for the implicit method. The implicit equation is solved by Newton's method, and the floating point state $X^{it}(t+\tau)$ is rounded to the nearest integer values.

QSSA

The explicit and implicit tau-leaping methods achieve increased computational efficiency by attempting to leap over many fast reactions. The quasi-steady state approximation (QSSA) [9] improves the efficiency by relying on this steady state assumption: the net rate of formation is approximately equal to zero when the fast reacting species are in steady state. Consider common enzyme kinetic reactions using Michaelis-Menten kinetics. For substrate S, enzyme E, and product P, the Michaelis-Menten reaction is

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} E:S \xrightarrow{k_2} P + E,$$

where k_1, k_{-1}, and k_2 are the rate constants. E:S is the enzyme-substrate complex after the combination of substrate and enzyme. The rate equations corresponding to this reaction are

$$\begin{aligned} \frac{d[S]}{dt} &= -k_1[S][E] + k_{-1}[E:S], \\ \frac{d[E:S]}{dt} = -\frac{d[E]}{dt} &= -(k_{-1} + k_2)[E:S] + k_1[S][E], \\ \frac{d[P]}{dt} &= k_2[E:S], \end{aligned}$$

where $[X]$ denotes the concentration of the species X. Assume that total enzyme concentration $E_T = [E] + [E:S]$ and $[S] \gg [E:S]$. From the assumption, the characteristic time scale of $[S]$ is very slow in comparison to that of $[E:S]$, and $[E:S]$ reaches steady state quickly. From the quasi-steady state assumption, the rate equation for $[E:S]$ is approximated by

$$\frac{d[E:S]}{dt} = 0.$$

Mathematically, it is possible to obtain a single rate equation,

$$\frac{d[S]}{dt} = -k_2[E:S] = -\frac{k_2 E_T[S]}{K_m + [S]},$$

where $K_m = (k_{-1} + k_2)/k_1$.

Similarly, the quasi-steady state assumption can be applied to the stochastic formulation. The QSSA in stochastic kinetics implies that the net rate of change for the conditional probability distribution of the fast reacting species is equal to zero. For the above Michaelis-Menten kinetics, the separate master equation for the enzyme-substrate complex is

$$\frac{dP([E:S] \mid [\hat{S}]; t)}{dt} = 0,$$

where $[\hat{S}] = [S] + [E:S]$. From the above equation, one can easily derive the reduced system

$$\hat{S} \longrightarrow P,$$

with the propensity function

$$a(s) = \frac{k_2 E_T[\hat{S}]}{K_m + [\hat{S}]}.$$

Finally, the SSA is applied to this reduced system, and shows improved computational efficiency compared with using SSA on the original system.

tQSSA

In the previous section, the quasi-steady state approximation (QSSA) eliminates the fastest reacting variable under some assumptions. In Michaelis-Menten kinetics, the necessary condition for the QSSA is $S_0 \gg E_T$, where S_0 is the initial substrate concentration, and E_T is the total enzyme concentration. In a protein interaction network (PIN), however, the enzymes and substrates often swap their roles [16]. Therefore, the QSSA condition will not be true for such a PIN. Borghans et al. [17] proposed that the proper slow timescale variable is $[\hat{S}] = [S] + [E:S]$ instead of $[S]$. In terms of this variable, the deterministic equations are

$$E_T = [E] + [E:S] = \text{constant},$$
$$[E:S]^2 - \left(E_T + K_M + [\hat{S}]\right)[E:S] + E_T[\hat{S}] = 0,$$
$$\frac{d[\hat{S}]}{dt} = -k_2[E:S].$$

This is called the total quasi-steady state approximation (tQSSA). To derive the equations for the stochastic simulation under the tQSSA, reduce the system to

$$\hat{S} \longrightarrow P$$

with the propensity function

$$a(s) = k_2[E:S],$$

where

$$[E:S] = \frac{\left(E_T + K_m + [\hat{S}]\right) - \sqrt{\left(E_T + K_m + [\hat{S}]\right)^2 - 4E_T[\hat{S}]}}{2}.$$

Finally, the SSA is applied to this reduced system, and shows improved computational efficiency compared with using SSA on the original system. Moreover, the tQSSA overcomes the modeling shortcomings of the QSSA.

A Stochastic Cell Cycle Model

The molecular machinery of eukaryotic cell cycle control is known in more detail for budding yeast, *Saccharomyces cerevisiae*, than for any other organism. Therefore, the unicellular budding yeast is an excellent organism for which to study cell cycle regulation. Molecular biologists have dissected and characterized individual cell cycle components and their interactions to derive a consensus picture of the regulatory network of the budding yeast. Figure 1 shows the wiring diagram for the budding yeast model [13]. The diagram should be read from the bottom-left toward the top-right. Solid arrows represent biochemical reactions, and dashed lines represent how components may influence one another. Empirical

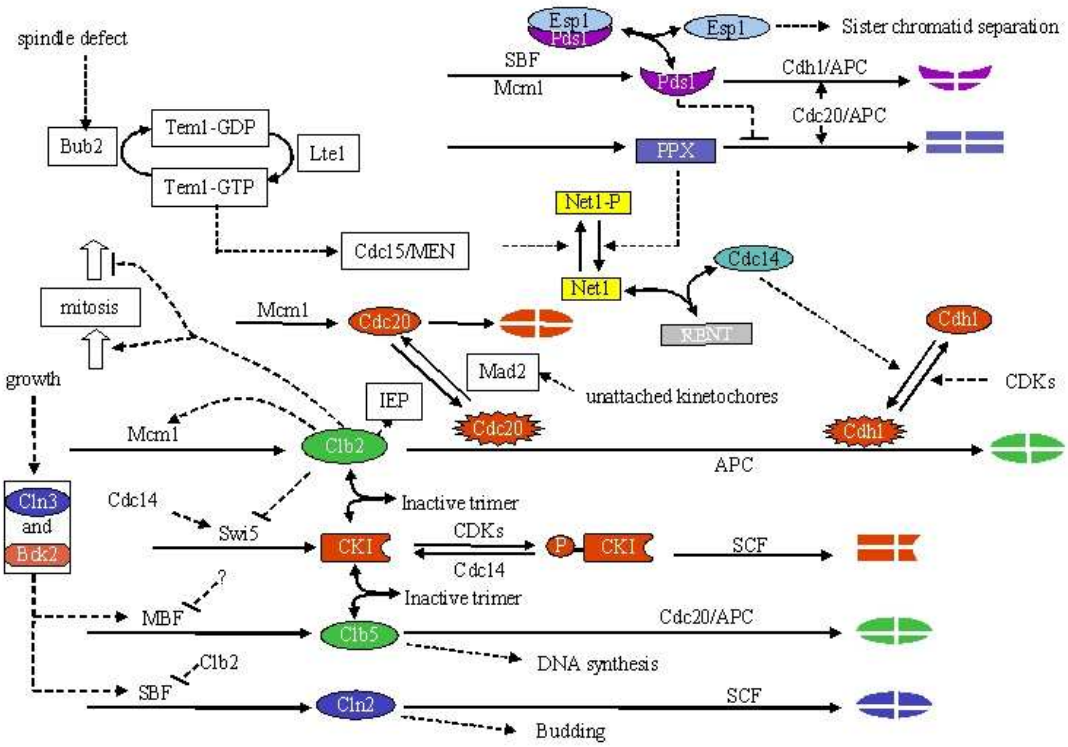

Figure 1. Wiring diagram of budding yeast.

results of simulation runs for a stochastic formulation of the budding yeast model are presented in the following section.

Each run of the stochastic simulation provides potentially different results, and the real goal is to deduce the true population distribution for the potential outcomes. This requires potentially many thousands of simulation runs. Each simulation run requires different amounts of processor time. When hundreds of simulation runs are assigned statically to a processor, the total simulation times can be quite different across the processors, causes a serious load imbalance. A dynamic load balancing algorithm was developed that more evenly distributes work to the processors.

The basic idea of the load balancing algorithm is a master/slave paradigm that dynamically adjusts the task chunk size. Pseudocode (with constants tuned for the problem at hand) for the load balancing follows.

Algorithm GetTask(n, p, task)

1. ▷ **Input:** n = number of remaining tasks,
2. p = number of processors
3. ▷ **Output:** *chunk* = number of tasks for a processor
4. **begin**
5. setPoint ← floor (n/p)
6. **if** $setPoint > 99$ **then**

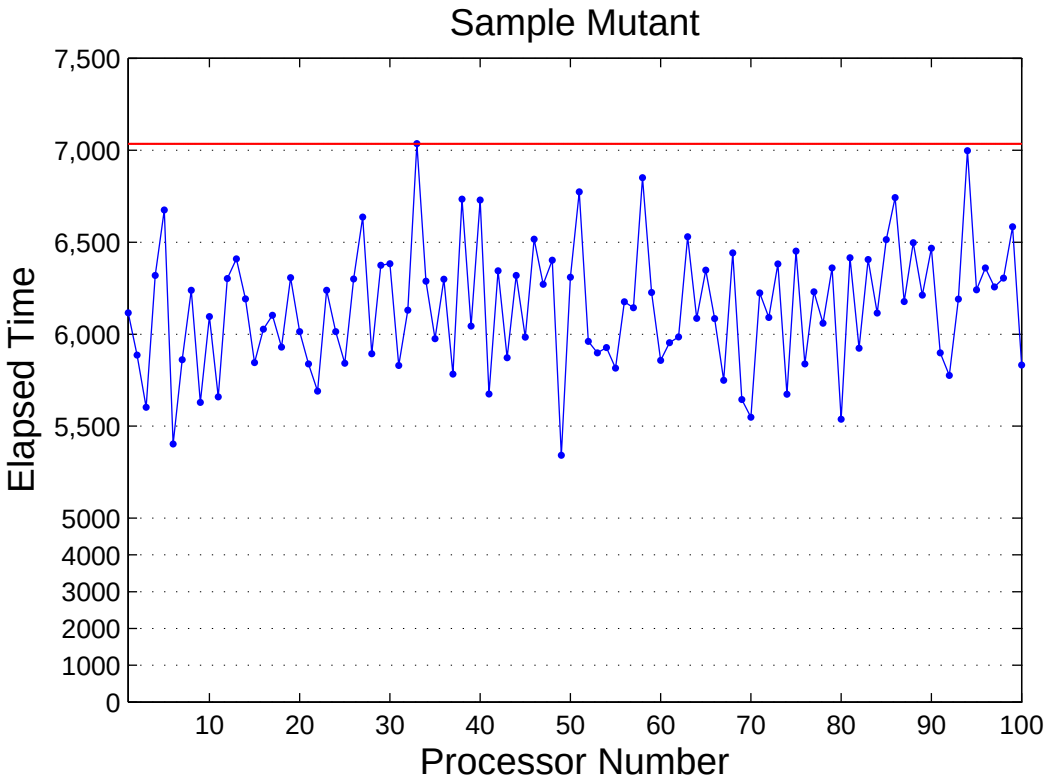

Figure 2. Static workload distribution.

7. $chunk \leftarrow$ floor $(setPoint \times 0.8)$
8. **else if** $setPoint > 5$ **then**
9. $chunk \leftarrow$ floor $(setPoint \times 0.5)$
10. **else then**
11. $chunk \leftarrow 1$
12. **end if**
13. **end**

The idea of varying the task chunk size as the size of the task queue decreases is well known in parallel computing, where it is called guided self-scheduling. Figures 2 and 3 clearly show the advantage of the load balancing algorithm. Using Virginia Tech's 2200 processor (2.3 GHz PowerPC 970FX) System X, 100 worker processors were used for 10,000 stochastic simulations for a budding yeast prototype double mutant. Figure 2 shows the time that each processor required to complete its assigned tasks. The variance between processors is high. Figure 3 shows processor times using the dynamic workload distribution. Here, the variance in processor times is negligible, even though individual runs of the simulation have large fluctuations in their cell cycles.

Tables 1 and 2 show the average and total times for the static and dynamic workload distributions. It is observed that dynamic workload distribution reduces system resource use by approximately 14%.

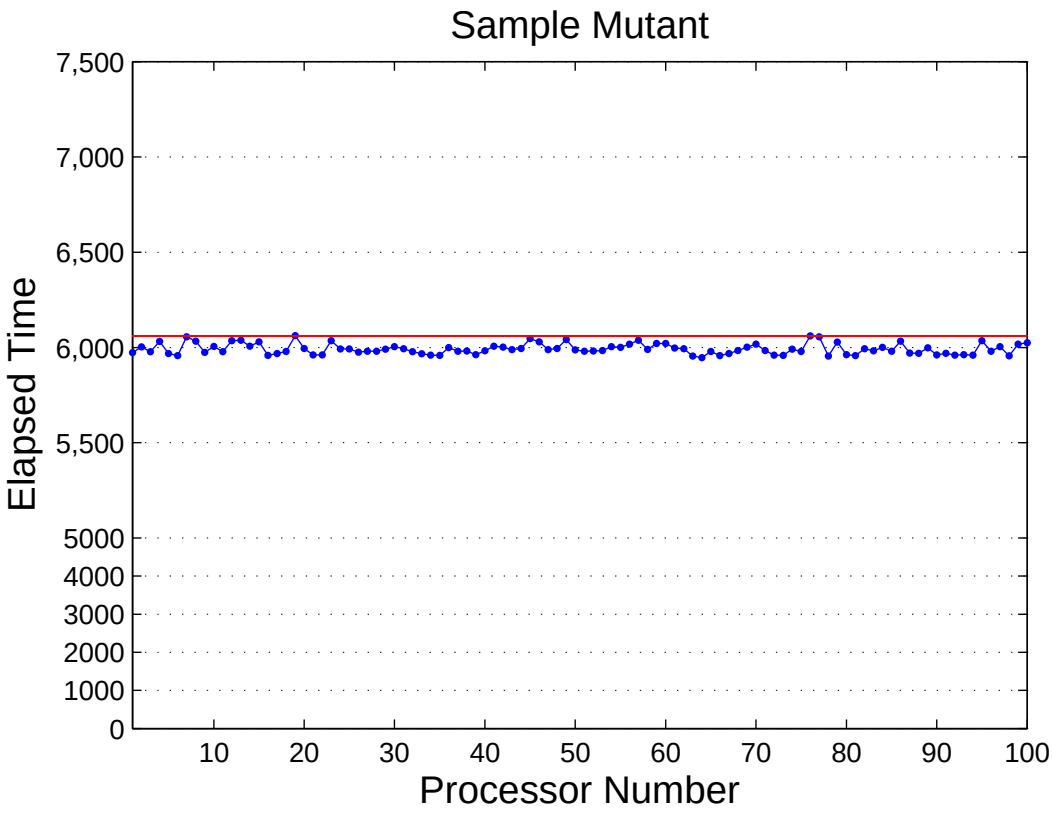

Figure 3. Dynamic workload distribution.

Table 1. Execution times for static distribution.

Parallel Runtime	Time (seconds)
T_p(average)	6144.93
T_p(max)	7035.95
Wall clock time	**7040.72**

Table 2. Execution times for load balanced distribution.

Parallel Runtime	Time (seconds)
T_p(average)	5992.43
T_p(max)	6061.78
Wall clock time	**6066.84**

Numerical Experiments for Simple Reactions

The irreversible isomerization system and the Goldbeter-Koshland switch are used here to compare various stochastic algorithms. The irreversible isomerization shows the computational efficiency of the explicit tau-leaping method compared with the SSA on a nonstiff system. The GK switch is a suitable stiff model to compare various stochastic algorithms (see, e.g., [18]).

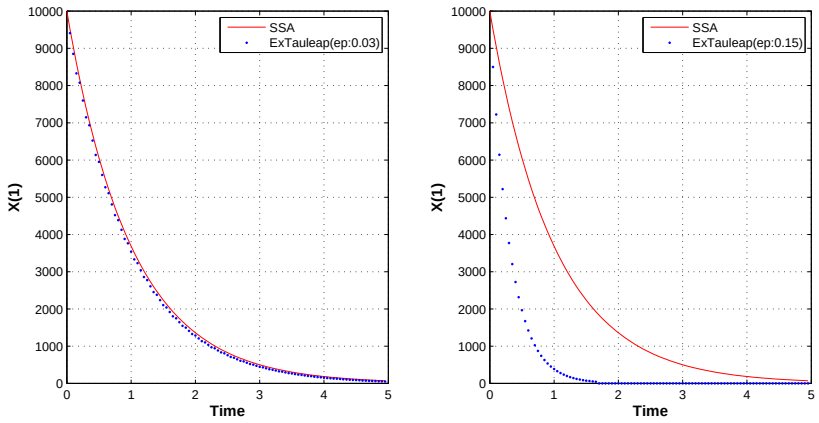

Figure 4. The SSA simulation (solid lines) and the explicit tau-leaping simulation (dotted lines) for the irreversible isomerization. The error control parameter ϵ is 0.03 (left) and 0.15 (right).

Table 3. The number of runs and elapsed CPU time (sec) with the SSA and explicit tau-leaping method, where $t_f = 5$, $c_1 = 1$, $X_1 = 10^4$, and $\epsilon = 0.03$.

Number of runs	1000	5000	10000	50000
SSA	48.74	240.40	487.76	2433.28
Explicit Tau-leaping	2.73	14.21	28.37	143.29

Irreversible isomerization

The first application is the simplest chemical reaction, the irreversible isomerization

$$S_1 \xrightarrow{c_1} 0.$$

The initial parameters for this system are reaction rate constant $c_1 = 1$, and 10^4 S_1 molecules at time 0. Figure 4 shows the SSA and explicit tau-leaping results. The final time is $t_f = 5$ and the error control parameters are $\epsilon = 0.03$ (left) and $\epsilon = 0.15$ (right). In order to compare the exactness, 5000 runs are used for both methods, with Fig. 1 showing mean and mean $\pm$ one standard deviation. These three lines in the figure are visually identical. Figure 1 shows that increasing ϵ, while making tau-leaping faster, affects the accuracy. The exact SSA method requires 9925 steps to reach the final time ($t_f = 5$). The tau-leaping method requires 167 (34) leap steps for $\epsilon = 0.03$ ($\epsilon = 0.15$). Table 3 compares the computational efficiencies of the explicit tau-leaping method and the SSA. Explicit tau-leaping is about 17 times faster than the SSA with an appropriate approximation for this model.

Goldbeter-Koshland switch

The Goldbeter-Koshland switch (GK switch) consists of a substrate-product pair (S and S_p) that is interconverted by two enzymes (E and D):

$$D + S_p \underset{k_{-1d}}{\overset{k_{1d}}{\rightleftarrows}} D : S_p \xrightarrow{k_{2d}} S + D,$$

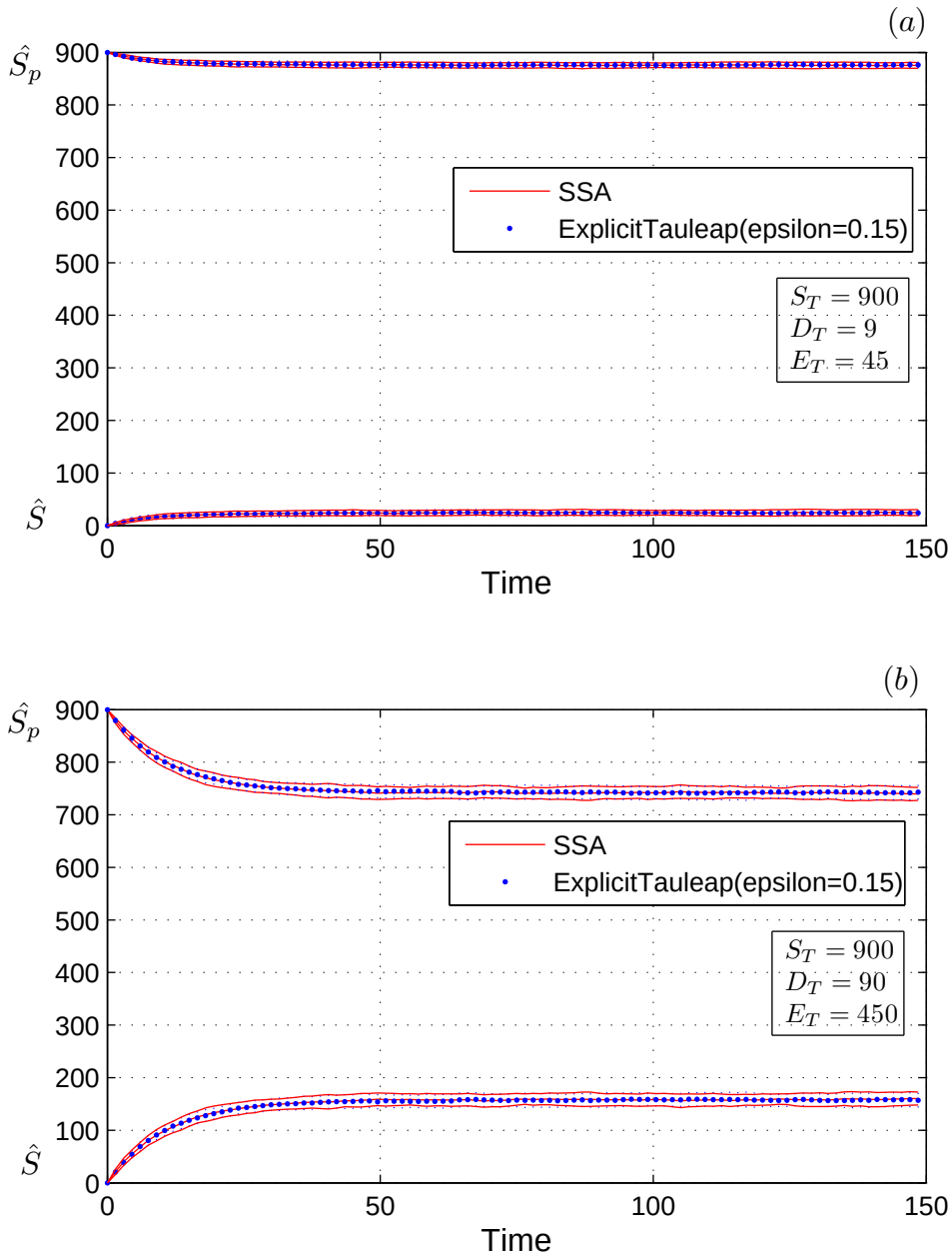

Figure 5. The SSA simulation (solid lines) and the explicit tau-leaping simulation (dotted lines) for the GK switch. State space (S_T, D_T, E_T) is (a):(900, 9, 45) and (b):(900, 90, 450).

$$E + S \underset{k_{-1e}}{\overset{k_{1e}}{\rightleftarrows}} E : S \xrightarrow{k_{2e}} S_p + E.$$

The parameter values are $S_T = 900$, $k_{1d} = 0.05555\text{min}^{-1}$, $k_{-1d} = 0.83\text{min}^{-1}$, $k_{2d} = 0.17\text{min}^{-1}$, $k_{1e} = 0.05\text{min}^{-1}$, $k_{-1e} = 0.8\text{min}^{-1}$, and $k_{2e} = 0.1\text{min}^{-1}$. In order to observe how the relationship between S_T and E_T affects the results, the two cases $(D_T, E_T) = (9, 45)$ and $(D_T, E_T) = (90, 450)$ are compared. $\hat{S}_p$ and $\hat{S}$ are defined by $\hat{S}_p = S_p + D : S_p$ and $\hat{S} = S + E : S$.

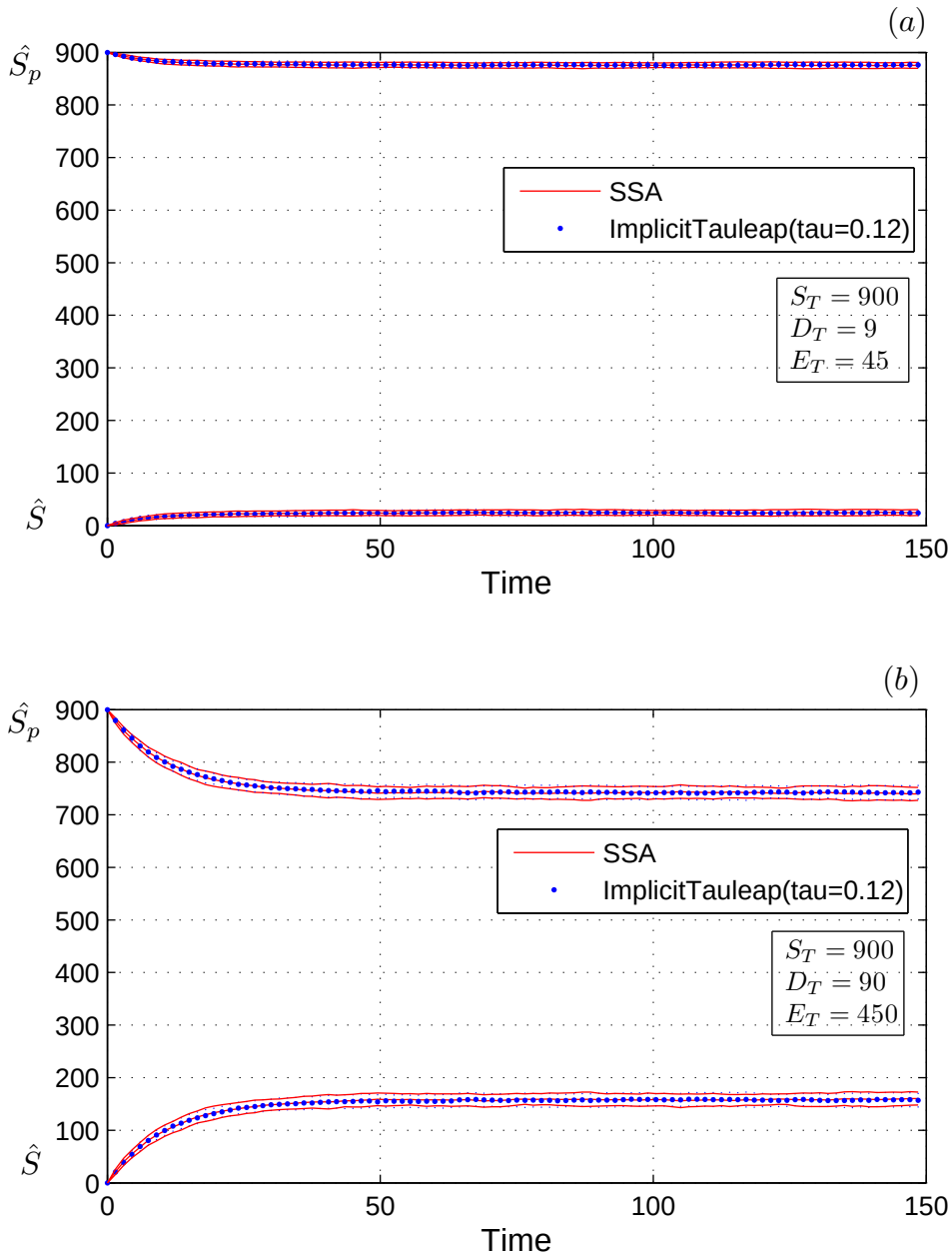

Figure 6. The SSA simulation (solid lines) and the implicit tau-leaping simulation (dotted lines) for the GK switch. State space (S_T, D_T, E_T) is (a):(900, 9, 45) and (b):(900, 90, 450).

Figures 5 and 6 show the mean and mean $\pm$ one standard deviation trajectories of $\hat{S}_p$ and $\hat{S}$ for the explicit and implicit tau-leaping approximation algorithms and the SSA. When $\hat{S}_p$ reaches steady state, the standard deviations of the SSA are approximately 4.3 (a) and 13.2 (b). In the explicit method, the error parameter value ($\epsilon = 0.15$) was chosen so that the standard deviation matched that of the SSA. The fixed step value ($\tau = 0.12$) was chosen similarly for the implicit method. Table 4 shows that the explicit method

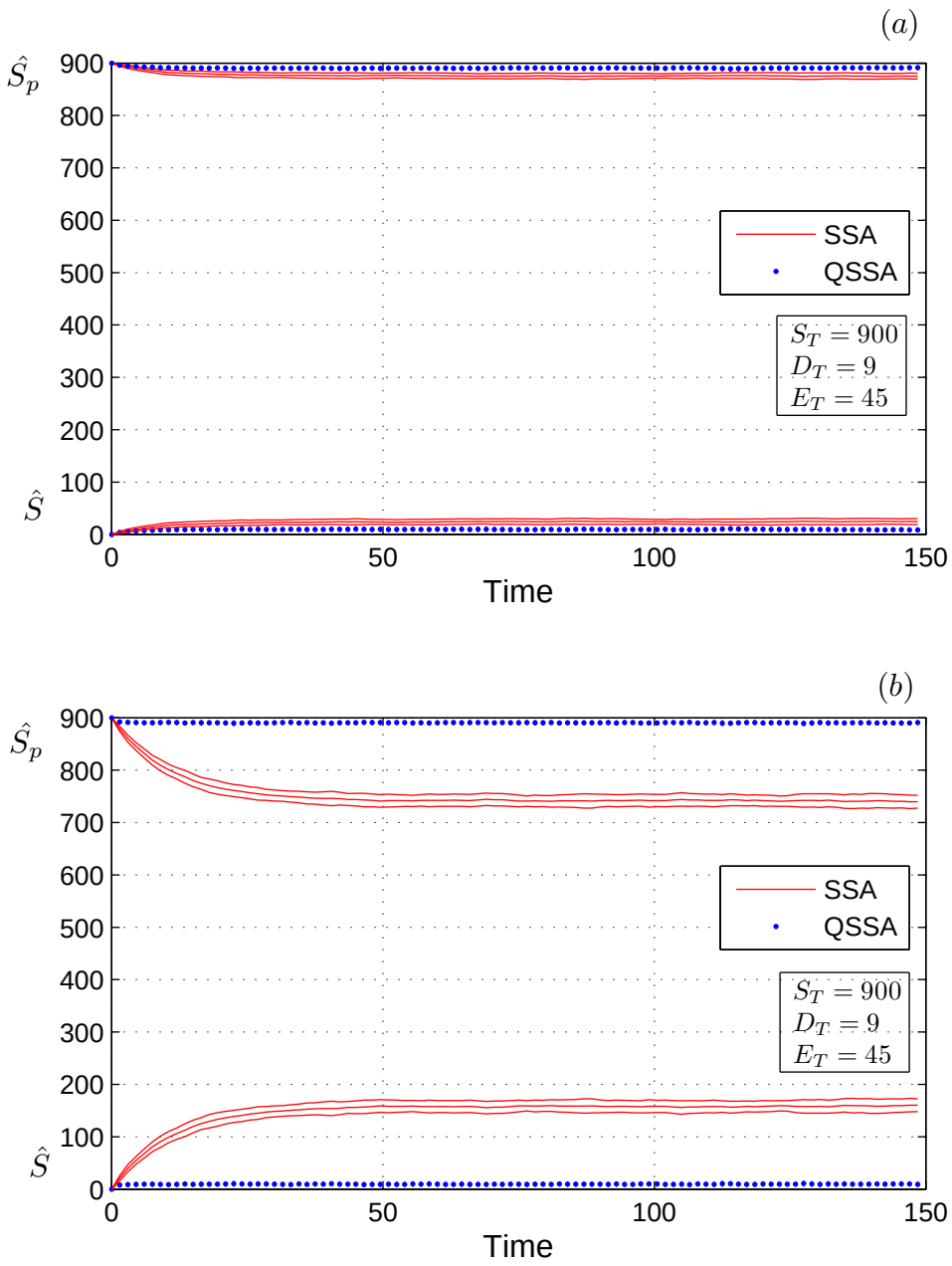

Figure 7. The SSA simulation (solid lines) and the QSSA simulation (dotted lines) for the GK switch. State space (S_T, D_T, E_T) is (a):(900, 9, 45) and (b):(900, 90, 450).

is faster than the implicit method for the same accuracy, due to the cost of the Newton iteration in the implicit method.

Figures 7 and 8 show the mean and mean $\pm$ one standard deviation trajectories of $\hat{S}_p$ and $\hat{S}$ for the QSSA and tQSSA algorithms. Figure 7(b) shows that the results with the QSSA are different from those for the SSA. The explanation in the QSSA section implies that the QSSA requires $S_T \gg E_T$. If $S_T \approx E_T$ or

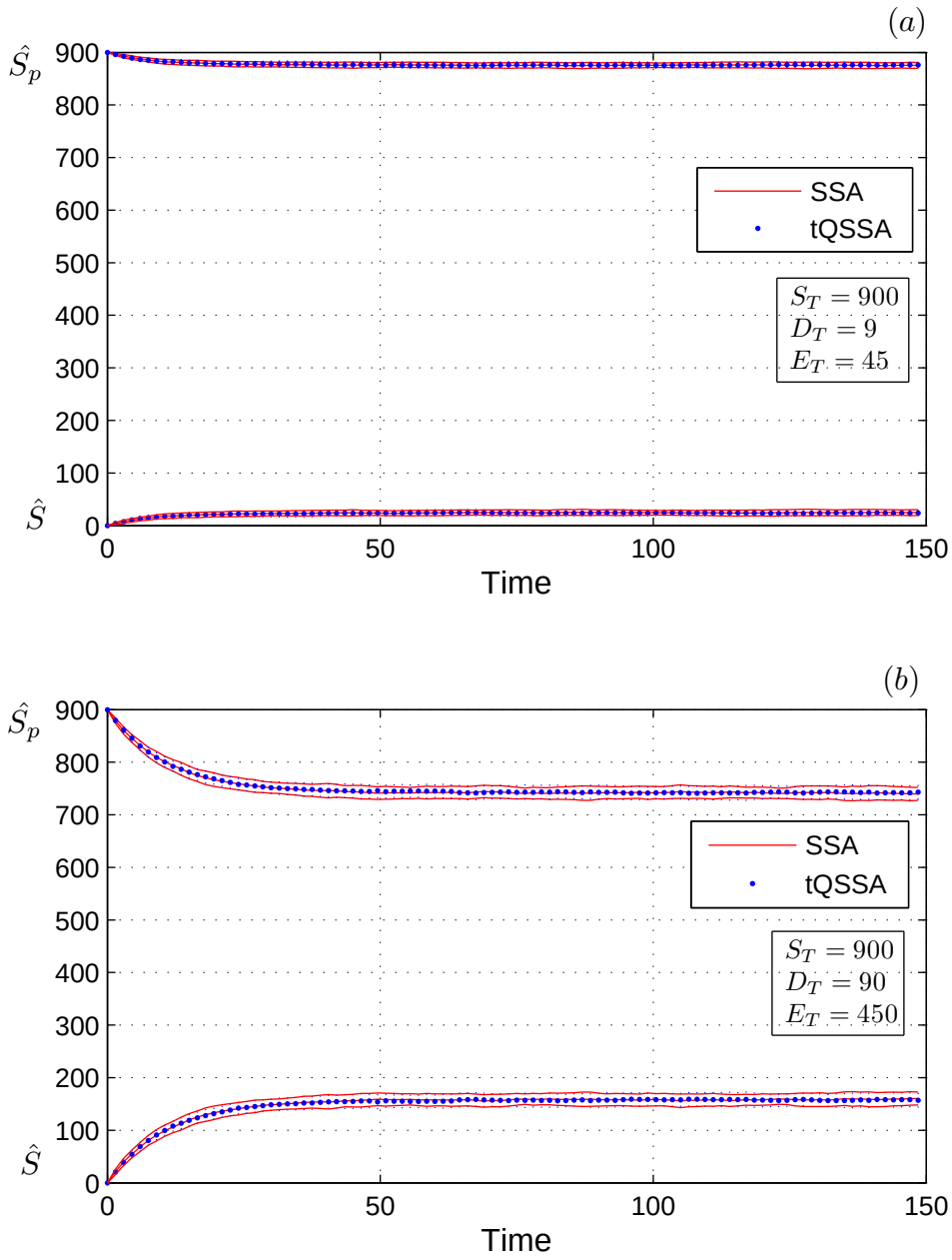

Figure 8. The SSA simulation (solid lines) and the tQSSA simulation (dotted lines) for the GK switch. State space (S_T, D_T, E_T) is (a):(900, 9, 45) and (b):(900, 90, 450).

$S_T \ll E_T$, then the results from the QSSA algorithm are not reliable. Figure 7(a) shows that if $S_T \gg E_T$, then the results of the QSSA algorithm are similar to those from the SSA. In contrast with Figure 7, Figure 8 shows that the tQSSA algorithm works when $S_T \approx E_T$ or $S_T \ll E_T$.

In terms of CPU time, Table 4 shows that the QSSA and tQSSA algorithms are the fastest approximate algorithms. The QSSA and tQSSA algorithms are almost 20 times faster than the SSA. The explicit and

Table 4. The number of runs and elapsed CPU time (sec) for the SSA, explicit tau-leaping, implicit tau-leaping, QSSA, and tQSSA algorithms.

Number of runs	1000	5000	10000	50000
SSA	44.57	216.77	434.10	2175.45
Explicit Tau-leaping	4.75	18.23	46.57	181.31
Implicit Tau-leaping	18.43	89.13	181.29	887.44
QSSA	2.63	13.23	26.33	132.47
tQSSA	2.67	13.33	26.79	133.68

implicit tau-leaping approximations also have improved computational time over the SSA.

Cell Cycle Results

Event implementation

Stochastic methods require the model to be in terms of population because they consider reactions with individual molecules. Because the original budding yeast model is based on normalized concentration values, a conversion process from an ODE model into a model in terms of number of molecules is needed. The conversion process is done using JigCell [12], and consists of two phases, *unit checking* and *model conversion*. Unit checking verifies physical unit consistency inside the model. Model conversion converts the model by changing values of species and parameters based on the unit information.

After creating the population-based budding yeast model, there is technical issue that must be addressed. In SBML, an *event* is triggered when some condition is met. There are events defined to divide the cell or mark checkpoints within the cell cycle stages. A typical deterministic event has the form:

```
if (X > threshold)
  then (Event is triggered)
```

Because of the random nature of stochastic simulation, as illustrated in Figure 9, unwanted events can be triggered when a deterministic SBML event is used for the stochastic model. In Figure 4, an unwanted event (B) can be triggered with a wanted event (A) by using a deterministic event handling equation.

To prevent unwanted events, the event logic has to be rewritten to tolerate the situation where the value of X oscillates around the threshold. A second threshold value can be defined from the threshold and the direction of the test (greater than or less than). For budding yeast, this second threshold equals .5*threshold (for a greater-than test) or 1.5*threshold (for a less-than test). For instance, the event code above would be changed to:

```
if (X < second threshold)
  then (EventFlag ← TRUE)
if (X > threshold AND EventFlag = TRUE)
  then (event is triggered;
```

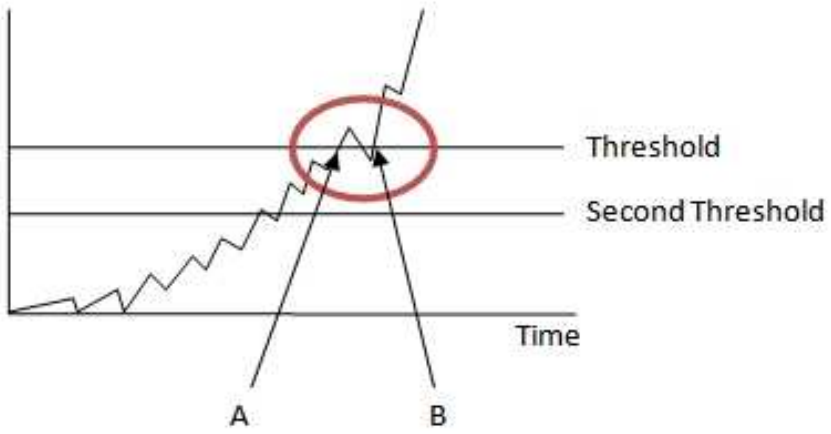

Figure 9. Event handling.

`EventFlag` ← `FALSE`)

StochKit [11] was used to do stochastic simulation of the converted budding yeast model, using the SSA option for the most precise results. JigCell can generate the StochKit model file by using the population based budding yeast model file.

Wild type simulation results with the SSA

To compare the stochastic results with deterministic cell cycle simulation, mass and several representative species' trajectories are shown in Figures 10 and 11. The deterministic result is from the XPP ODE simulator using JigCell. For comparison, the stochastic simulation results are converted back to normalized concentrations.

Table 5. Execution time comparisons with different methods for wild type budding yeast.

Method	Time(seconds)	The Number of Steps
Direct SSA	25.03	2,495,000
Explicit Tau-leaping(1) with $\epsilon = 0.01$	82.893	2,444,000
Explicit Tau-leaping(1) with $\epsilon = 0.05$	312.07	1,232,000
Explicit Tau-leaping(1) with $\epsilon = 0.15$	618.41	290,000
Explicit Tau-leaping(2) with $\epsilon = 0.01$	28.90	2,442,000
Explicit Tau-leaping(2) with $\epsilon = 0.05$	29.44	2,319,000
Explicit Tau-leaping(2) with $\epsilon = 0.15$	29.16	1,835,000
Implicit Tau-leaping with $\tau = 0.001$	7965.18	500,000
Implicit Tau-leaping with $\tau = 0.005$	1920.34	100,000
Implicit Tau-leaping with $\tau = 0.010$	1012.87	50,000

Wild type results comparisons

Table 5 shows the results of the stochastic budding yeast simulation with direct SSA, explicit tau-leaping, and implicit tau-leaping methods. The final time is $t_f = 500$. In order to select the largest

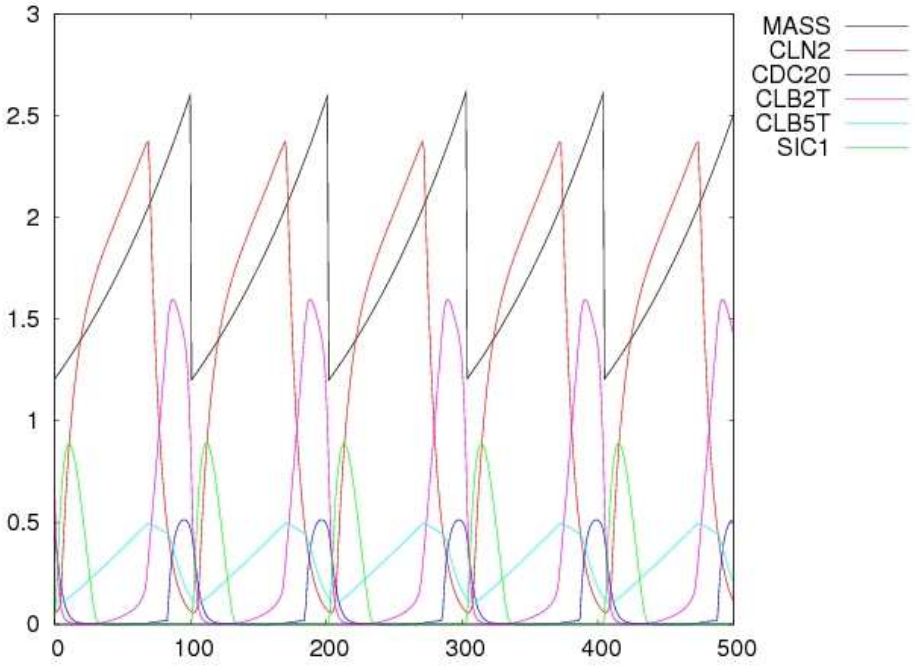

Figure 10. Deterministic cell cycles.

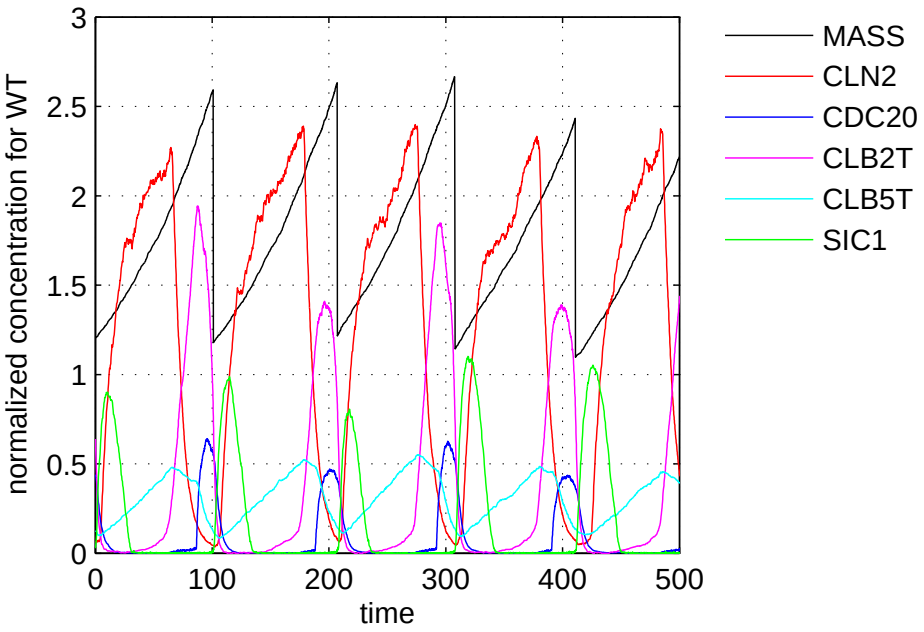

Figure 11. Stochastic cell cycles.

value of τ that satisfies the leap condition, the Jacobian matrices for the propensity functions are used for the explicit tau-leaping(1) method [14]. Another approach, explicit tau-leaping(2), is to select τ such that relative changes in the propensity functions are bounded [15]. The fixed step values were chosen similarly for the implicit method.

For direct SSA, the average time to run the simulation is approximately 25 seconds with 2,495,000 steps. From the results of simple chemical reactions, the explicit and implicit tau-leaping methods improved computational time over the SSA while obtaining equally accurate simulation results. However, in the budding yeast model, the tau-leaping methods required substantially more simulation time. The Jacobian matrices for the propensity functions are used for explicit tau-leaping(1), and the time was increased for reduced steps. Thus the computational time for handling Jacobian matrices is larger than the computational time for steps. Even though the new tau selection method was used, the computational time did not improve over that of the SSA. Implicit tau-leaping took more time than explicit tau-leaping. Therefore, direct SSA is the most effective for the budding yeast model, and mutants in the next section were tested with the direct SSA method.

Mutants

For the mutants considered in this paper, stochastic simulations show that often they neither divide endlessly nor fail to divide at all, meaning that they divide for several cycles, then stop dividing. The number of cycles until stopping division is different for different trajectories. To generate enough data to perform a statistical analysis, 10,000 independent simulations are executed from the same initial point using the load balancing parallel algorithm.

Let the random variable X denote the number of cell divisions before the cell stops dividing, and assume that the probability p of not dividing is constant and independent of the cell's previous history. Then X has a modified geometric distribution given by

$\mathrm{P}(X = n) = p(1-p)^n$ where $n = 0, 1, ...$

Further, the probability of having n or more cycles is given by

$\mathrm{P}(X \geq n) = (1-p)^n$ where $n = 0, 1, ...$

If either probability is plotted on a log scale against n, it will be a straight line with a slope of $\log(1-p)$. In the analysis of the mutant simulations, a best least squares fit straight line is used to extract the slope and estimate the value of p.

In wet lab experiments, the viability of mutants is assessed by determining whether single mutant cells could grow into a colony. To determine the relationship between viability and the probability of ceasing to divide, simulation is used to determine the probability of one cell producing a colony of size greater than 10^6 in 32 division cycles. 32 division cycles correspond roughly to incubating the cells on a plate for two days. A 1 mm^3 colony has about 20×10^6 cells. Therefore, it is assumed that 0.05 mm^3 can be visible. The results of these simulations are summarized in Figure 12. From this figure, the ability to observe colonies has a roughly switch-like characteristic with the switching point near $p = 0.25$.

It is interesting that some mutants have different fates in different growth media. For example, some mutants of budding yeast are inviable in "rich" medium (fast growth rate) but partially viable in "poor" medium (slow growth rate). "Partially viable" means that cells growing in poor medium have some

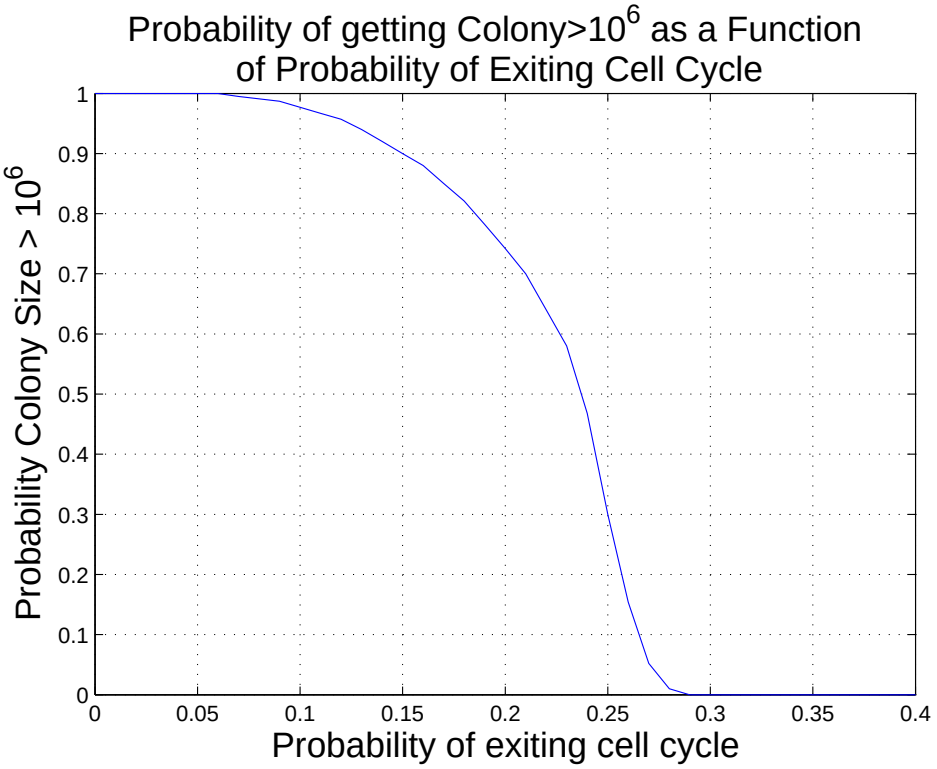

Figure 12. Probability of getting colony.

probability of not dividing ($0 < p < 0.5$, p as defined above). In rich medium, $p = 1$. Partial viability is not an option in a deterministic model, for which all cells behave identically (either $p = 0$ or $p = 1$).

Simulation results for a prototype model of a partially viable, budding yeast mutant are reported in Figures 13 and 14. For fast growth rates, every simulated cell arrests in the cell cycle ($p = 1$). If the growth rate is sufficiently slow, then simulated cells begin to divide with a certain probability $1 - p$. For the case in Figure 14, $p = 0.22$. From Figure 12, this implies that the probability of a single cell forming a colony is 65%, which could be compared to experimental observations for a particular mutant. The point is that stochastic simulation accounts for the property of partial viability that a deterministic simulation cannot.

Conclusions and Future Work

The simulation results reported here, while limited, show important characteristics of each approximation algorithm. The explicit tau-leaping method improves computational efficiency, compared to the SSA, for nonstiff systems, but can be unstable on stiff systems. The implicit tau-leaping method is stable, but much slower than the explicit method. The tQSSA algorithm produces excellent agreement with the SSA and is more efficient by an order of magnitude. Some of these approaches are simulated with the budding yeast model, and were shown to be impractical for this realistic cell cycle model.

The budding yeast stochastic simulation results reported here, while limited, show important characteristic aspects of cell cycle empirical data, such as mixed mutant viability. Because random

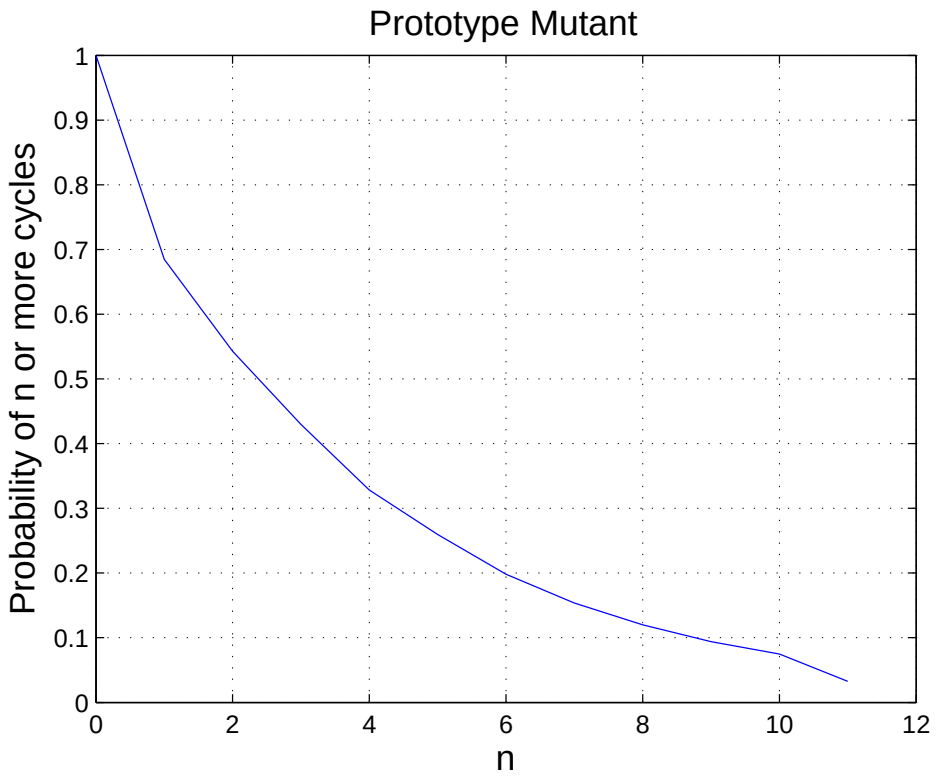

Figure 13. $P(X \geq n)$.

fluctuations are important to accurately simulate mutants, some major regulatory proteins occur in small numbers, the stochastic approach is more realistic and accurate than the deterministic approach for modeling the budding yeast cell cycle. The guided self-scheduling load balancing algorithm is effective for managing the large numbers of SSA trajectories required by the stochastic approach.

The data collected here are only the first step toward calculating population doubling times, and probabilities for successful colony formation of the various mutants. Those probabilities are one of the tests for verifying that the simulations match experimental results. The results are also perhaps skewed by the fact that all simulation runs begin with a static initial condition that might not be representative of the true population. In future work, the authors will calculate population statistics and use them to generate appropriate initial conditions.

REFERENCES

[1] Murray A, Hunt T: *The Cell Cycle. An Introduction.* USA: Oxford University Press; 1993.

[2] Chen KC, Calzone L, Csikasz-Nagy A, Cross FR, Novak B, Tyson JJ: **Integrative analysis of cell cycle control in budding yeast.** *Mol. Biol. Cell* 2004, **15**:3841–3862.

[3] Gillespie DT: **A general method for numerically simulating the stochastic time evolution of coupled chemical reactions.** *Journal of Computational Physics* 1976, **22**:403–434.

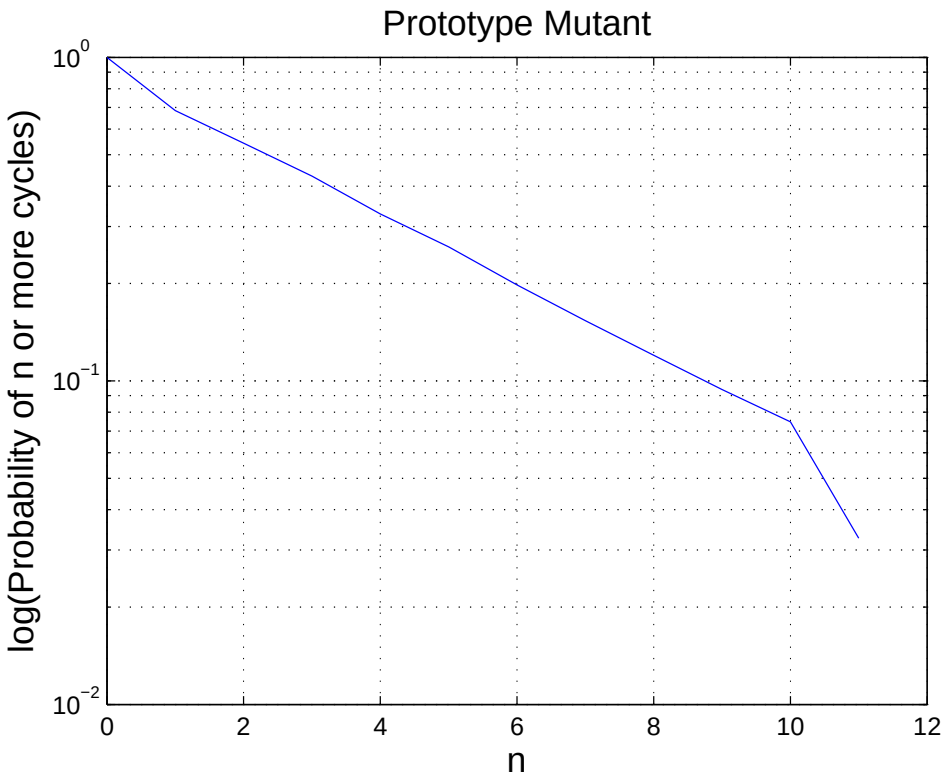

Figure 14. $\log(P(X \geq n))$

[4] Gillespie DT: **Exact stochastic simulation of coupled chemical reactions.** *Journal of Physical Chemistry* 1977, **81**:2340–2361.

[5] Gibson MA, Bruck J: **Efficient exact stochastic simulation of chemical systems with many species and many channels.** *Journal of Physical Chemistry* 2000, **104**:1876–1889.

[6] Gillespie DT: **Approximate accelerated stochastic simulation of chemically reacting systems.** *Journal of Chemical Physics* Journal of Chemical Physics. 2001, **115**:1716–1733

[7] Cao Y, Gillespie DT, Petzold LR: **The slow-scale stochastic simulation algorithm.** *Journal of Chemical Physics* 2005, **122**:014116+.

[8] Rathinam M, Petzold LR, Cao Y, Gillespie DT: **Stiffness in stochastic chemically reacting systems: the implicit tau-leaping method.** *Journal of Chemical Physics* 2003, **119**:12784–12794.

[9] Rao CV, Arkin AP: **Stochastic chemical kinetics and the quasi-steady-state assumption: application to the Gillespie algorithm.** *Journal of Chemical Physics* 2003, **118**:4999–5010.

[10] Cilibetro AC, Capuani F, Tyson JJ: **Modeling networks of coupled enzymatic reactions using the total quasi-steady state approximation.** *PLoS Computational Biology* 2007, **3**:463–472.

[11] Li H, Cao Y, Petzold L, Gillespie D: **Algorithms and software for stochastic simulation of biochemical reacting systems.** *Biotechnology Prog.* 2008, **24**:56–61.

[12] Wang P, Randhawa R, Shaffer CA, Cao Y, Baumann WT: **Converting macromolecular regulatory models from deterministic to stochastic formulation.** In *Proceedings of the 2008 Spring Simulation Multiconference*, 2008, SpringSim '08. ACM, New York:385–392.

[13] Virginia Tech: The JigCell website, http://jigcell.biol.vt.edu

[14] Gillespie DT, Petzold LR: **Improved leap-size selection for accelerated stochastic simulation.** *Journal of Chemical Physics* 2003, **119**:8229–8234.

[15] Cao T, Gillespie DT, Petzold LR: **Efficient stepsize selection for the tau-leaping simulation.** *Journal of Chemical Physics* 2006, **124**:144109+.

[16] Blüthgen N, Herzel H: **How robust are switches in intracellular signaling cascades?.** *Journal of Theoretical Biology* 2003, **225**:293–300.

[17] Borghans JAM, de Bore RJ, Segel LA: **Extending the quasi-steady state approximation by changing variables.** *Bulletin of Mathematical Biology* 1996, **58**:43–63.

[18] Barik D, Paul MR, Baumann WT, Cao Y, Tyson JJ: **Stochastic simulation of enzyme-catalyzed reactions with disparate timescales.** *Biophys J.* 2008, **95(8)**:3563–74.