Stochastic Simulation Methods for Biochemical Systems with Multi-state and Multi-scale Features

Zhen Liu

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Application

Yang Cao, Chair

T. M. Murali

Adrian Sandu

Clifford A. Shaffer

Jianhua Xing

October 22, 2012

Blacksburg, Virginia

Keywords: SSA, Stochsim, rule-based modeling, QSSA, hybrid method

Stochastic Simulation Methods for Biochemical Systems with Multi-state and Multi-scale Features

Zhen Liu

(ABSTRACT)

In this thesis we study stochastic modeling and simulation methods for biochemical systems. The thesis is focused on systems with multi-state and multi-scale features and divided into two parts. In the first part, we propose new algorithms that improve existing multi-state simulation methods. We first compare the well known Gillespie's stochastic simulation algorithm (SSA) with the StochSim, an agent-based simulation method. Based on the analysis, we propose a hybrid method that possesses the advantages of both methods. Then we propose two new methods that extend the Network-Free Algorithm (NFA) for rule-based models. Numerical results are provided to show the performance improvement by our new methods. In the second part, we investigate two simulation schemes for the multi-scale feature: Haseltine and Rawlings' hybrid method and the quasi-steady-state stochastic simulation method. We first propose an efficient partitioning strategy for the hybrid method and an efficient way of building stochastic cell cycle models with this new partitioning strategy. Then, to understand conditions where the two simulation methods can be applied, we develop a way to estimate the relaxation time of the fast sub-network, and compare it with the firing interval of the slow sub-network. Our analysis are verified by numerical experiments on different realistic biochemical models.

Contents

List of Figures

List of Tables

Chapter 1

Overview

With the growing interest in more and more complicated biochemical systems, mathematical modeling and numerical simulation have become important and powerful tools for scientific research. In order to thoroughly understand the characteristics and dynamics of bio-chemical systems, researchers first construct mathematical models based on their best knowledge. Then numerical simulation is used as a validating tool for model modifications and, more importantly, later as an inexpensive and fast verification tool for scientific hypotheses. Many complex biological systems have been successfully modeled and simulated such as complex gene regulatory networks and metabolic pathways [3, 4, 5, 6]. Conventionally, models of biochemical systems are formulated as ordinary differential equations (ODEs) and simulated by deterministic and continuous methods. While these deterministic models correctly describe average behaviors of the modeled systems, the dynamics of these systems are actually stochastic in nature. In particular, in microscopic systems with small numbers of reacting

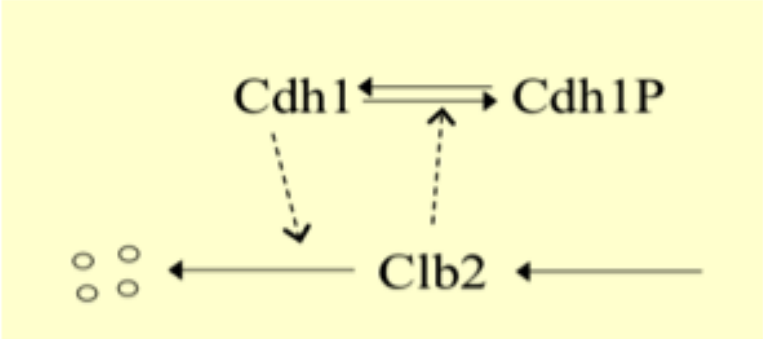


Figure 1.1: Module of bistable switch

molecules, the system dynamics may demonstrate discrete and stochastic features, which cannot be captured by conventional deterministic simulations. With critical species present in low populations of molecules, stochastic effects can be significant and sometimes determine the overall behaviors of biochemical systems [7, 8, 9, 10]. For example, a module of bistable switch, shown in Figure 1.1, is often found in biochemical networks.

It models a system with two steady states, each of which may represent a different biological phenotype. In deterministic simulations, the system always reaches one of the steady states, depending on the parameters and the initial conditions of the model, and stays there forever. However, in stochastic simulations, the system may reach both steady states owing to the stochastic nature. Thus both phenotypes, which have been observed in many biological wet-lab experiments, can also be realized in stochastic simulations. This module is a simple example of the important role the stochastic modeling and simulation play. Many much more complicated models can be found in literature [7, 8, 9, 10, 11, 12]. Therefore, in recent years more and more attention has been paid to stochastic modeling and simulation methods.

Gillespie's Stochastic Simulation Algorithm (SSA) [13, 14] is the most well-known method of stochastic simulation for chemically reacting systems. This method has been widely applied to different biochemical models and has been strongly believed to be the fundamental approach for stochastic simulations. One of the earliest stochastic models that used the SSA as its simulation tool is Arkin et al.'s λ-phage affected E-coli model [8]. This model successfully showed that an initially homogeneous E-coli cell population divides into different groups of phenotypic populations due to stochastic variations. Their simulation results were consistent with experimental observations and could not be produced by the conventional deterministic simulation. However, the SSA often exhibits low efficiency for many practical systems. Efficient stochastic simulation of biochemical systems presents a great challenge in computational science and motivates the development of many efficient algorithms in recent years (see Chapter 2). The goal of our research is to answer this challenge corresponding to different system features. We believe it is important to study the tradeoff between accuracy and efficiency of an algorithm together with specific system features present in complex biochemical models. In this thesis we are concerned with two important complexity features in biological systems: the *multi-state* feature and the *multi-scale* feature, which are known to cause low efficiency in the SSA.

The first part of this thesis deals with the efficiency challenge from systems with the multi-state feature. The multi-state feature usually arises when some chemical species can have multiple states with different chemical properties, for example, due to binding and unbinding, or (de-)phosphorylation of proteins. In systems with such feature, numbers of variables

and reactions in the SSA formulation may be very large and thus result in very low efficiency in simulation. A brief overview of the multi-state feature and our related research is given in Section 1.1. In this part, we first focus on the StochSim, one of the earliest stochastic simulation algorithms designed for multi-state models [15, 16, 17]. We provide a detailed comparison between the StochSim and the SSA, improve the StochSim, and propose a hybrid method by combining the two methods. Then we discuss another simulation method, recently proposed for multi-state systems and called the network-free algorithm (NFA) [18]. To improve its efficiency, we develop two new algorithms based on the NFA. We present numerical examples to demonstrate efficiency improvement of our new methods.

The second part of this thesis deals with the improvement of efficiency when a simulated model has the multi-scale feature, which is usually observed when a chemically reacting system has significant scale differences in reactions. Such difference can either exist in firing frequencies of reactions or in numbers of reactant molecules. One can refer to Section 1.2 for a brief introduction of the multi-scale feature and our related research. The SSA is usually slow on these models because it simulates each reaction firing, even though it is not necessary for those reactions with larger propensities or larger numbers of reactant molecules. Instead, these reactions often allow approximation or deterministic simulation. The approximation method leads to the SSA based on quasi-steady-state assumption (QSSA) [19] or partial equilibrium (slow scale SSA) [20, 21]. The deterministic method leads to the hybrid method proposed by Haseltine and Rawlings [22]. In this part we first study the hybrid method applied to a stochastic budding yeast cell cycle model with the multi-scale feature. We

apply Haseltine and Rawlings' hybrid method [22] on Kar et al.'s cell cycle model [11], and analyze the partitioning strategy for the best simulation efficiency. Based on the analysis, we propose a much simpler hybrid modeling approach to build stochastic cell cycle models compared to previous work. This new hybrid modeling approach applies not only to the stochastic cell cycle model, but also to other stochastic models where gene expression level is the major source for the stochastic effect. At the end of the second part, we try to study a system when both multi-state and multi-scale features are present. This is a bistable switch module based on multiple levels of phosphorylation, a core part in a new budding yeast cell cycle model developed by Barik et al. [12]. We will study the conditions, on which the quasi-steady-state assumption (QSSA) [19] and the Haseltine and Rawling's hybrid method are valid when applied to this simple bistable switch module. In this application, we analyze the accuracy and efficiency of these two methods and the proper conditions to apply these two methods by approximating the relaxation time of the fast subsystem. Numerical results are presented for this application.

In the following, we discuss the concepts of the multi-state feature, the simulation challenge it causes to the SSA, and an overview of our research in this topic in Section 1.1; Then we discuss the multi-scale feature in the cell cycle models followed by an overview of our research in this topic in Section 1.2.

1.1 The Multi-State Feature

In cellular regulatory systems, due to the multi-component composition of biologically reacting agents, proteins can bind and modify each other in many different ways. This is also known as the *multi-state* feature of biological systems. Multi-state species naturally arise from the possibility of a chemical species undergoing multiple levels of phosphorylation or methylation, or from the formation of complexes of proteins connected at various potential binding sites. Since different levels or different binding configurations lead to different biochemical properties, each variation must be represented as an individual species in the SSA formulation. The large number of potential bindings can yield a model of a very large size, and create the so-called combinatorial complexity [23, 24], which can result in great difficulty in model representation and low efficiency in stochastic simulations by the SSA. An example of the multi-state feature of a protein is illustrated in Figure 1.2. Protein A has three binding sites. The first one can bind to another kind of protein B, the second one can bind to protein C, and the third one is a phosphorylation site, which can bind to up to three phosphate groups. Since each site can also be empty (e.g. bound to nothing), the total number of states of protein A is $2 \times 2 \times 4 = 16$. Since each state of protein A has different chemical property, it has to be modeled by an individual state variable in the SSA formulation. Thus 16 state variables are needed to represent species A, and reactions based on species A may be repeated by 16 times.

Figure 1.2 only presents a simple example. A much more complicated example of the multi-

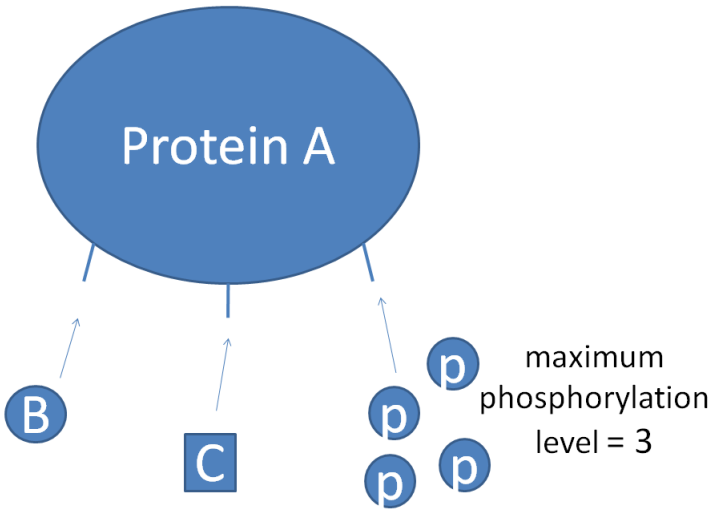


Figure 1.2: Example of multi-state feature of a protein

state feature is present in the modeling of bacterial chemotaxis, one of the most interesting and well-studied biological systems [25]. Figure 1.3 is a simplified diagram for the chemical regulatory pathway of bacterial chemotaxis [15, 16]. The core enzyme protein, called TTWWAA, can bind to 4 different kinds of proteins. It can also be phosphorylated and has multiple levels of methylations. In our simplified stochastic model of chemotaxis, the species TTWWAA has 180 different states, and the number of reactions is over 500.

Another set of examples with the multi-state feature are models of spatially inhomogeneous systems. As the SSA is based on the assumption of a spatially well-stirred system, its application on spatially inhomogeneous models requires discretizing the space into cells and duplicating the same set of variables and reactions in each cell. In this case, the resulting size of the system largely depends on the granularity of the discretization (e.g. the inverse of

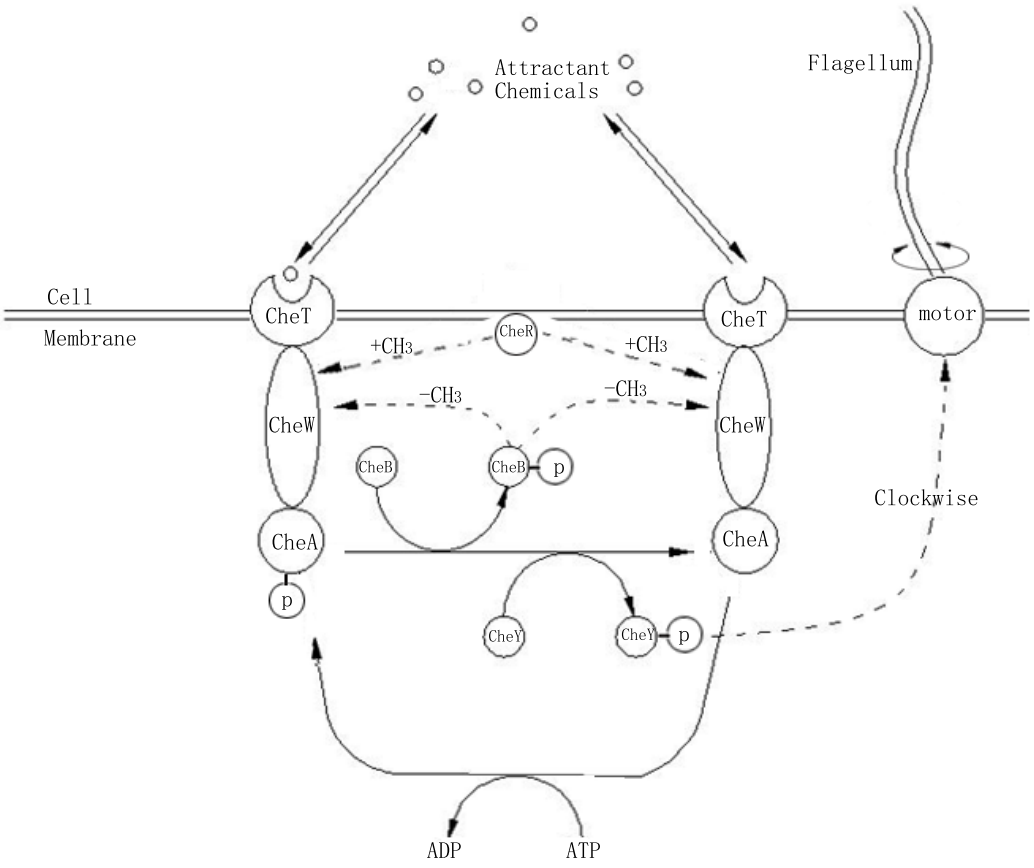


Figure 1.3: Chemical reacting network of bacterial chemotaxis. The dashed lines represent binding activities.

the total number of cells). In a recent caulobacter cell cycle model developed by Dr. Tyson's group at Virginia Tech, the appropriate number of cells is at least 100. To our estimation, their stochastic model in the SSA formulation will contain at least thousands of species and reactions. If we treat the spatial information (e.g. the index of the cell) of each species as the configuration of a special binding site, all these species can be considered as multi-state species.

From the above examples we can see that numbers of state variables and reactions increase dramatically owing to the multi-state feature. The computational complexities of the most widely-applied direct method and first reaction method in a single step are both linear to the number of reactions in the system [13, 14]. Although the best computational complexity achieved among different variations of the SSA in a single step is in logarithm of the number of reactions [26, 27, 28, 29], those algorithms usually have great overhead costs for special data structures, which have the same scale as the number of reactions. Therefore, the simulation efficiency of the SSA is greatly limited by the multi-state feature. In order to answer this challenge, *particle-based* simulation methods have been proposed [30, 31]. The StochSim, developed by Morton-Firth and Bray [15, 16, 17], is one of the earliest particle-based stochastic modeling and simulation methods. This method is specifically designed for models with the multi-state feature and is claimed to be equivalent to the SSA. We start with a fundamental question: Are the SSA and the StochSim really equivalent? In this thesis, we present a detailed comparison based on fundamental probability analysis. Complexity analysis and numerical experiments are presented. Our analysis reveals that the

probabilities calculated in the StochSim is a first-order approximation to those in the SSA. As a result, the StochSim requires a sufficiently small step size to maintain its accuracy. On the other hand, in the StochSim the step size is determined before simulation and can be very small when a system has a large total number of molecules. We conclude that the SSA is more efficient than the StochSim in most of the cases. However, when multi-state variables are involved in a system, the StochSim has its advantage. Also, as an attempt to develop efficient algorithms that combine the advantages of the population-based and particle-based methods, we propose the Hybrid SSA (HSSA), which is a variation of the SSA and incorporates the idea of particle-based simulation. This work has been published in IET Systems Biology [32].

From the comparison between the SSA and the StochSim we have learned that each method has its own advantages when simulating different types of models. However, there are many biochemical models that are not suitable for either the SSA or the StochSim. For example, a system may contain a multi-state species which has a large number of possible states, and at the same time may have a large total number of molecules. This kind of problems demand a new algorithm that can efficiently handle the multi-state situation with more flexibility than the StochSim. We propose an improved StochSim with an adaptive step size and removal of redundant objects. Then, we propose a stochastic hybrid method which combines the SSA and our improved StochSim. This method basically divides a system into two subsystems: One is simulated by the SSA and the other is simulated by the StochSim. Numerical experiments show that the hybrid method exhibits higher efficiency than both

the SSA and the StochSim. This work has been published in [33].

The Network-Free Algorithm (NFA) [18], another particle-based algorithm, was recently proposed to simulate rule-based models [34, 35] (see Chapter 2). The rule-based modeling method is specially designed to represent multi-state models. For systems with the multi-state feature, rule-based models are more convenient to build than the models in the SSA formulation. For a rule-based model, if the total number of molecules is small and many multi-state species are present, the NFA is more efficient than the SSA. However, when the total number of molecules increases, owing to the particle-based nature, the NFA becomes less efficient. We are particularly concerned with systems with the multi-state feature, as well as a large total number of molecules. We first compare the SSA and the NFA based on a group of test models. Through our analysis on the NFA, we find that a large portion of CPU time is spent on updating and maintaining the data structures during simulation. Therefore, we modify the NFA to treat single-state species with a population-based scheme. This changes NFA from a particle-based scheme to a hybrid scheme, called population-based NFA (PNFA) with reduced space and time complexity. To gain further improvements, we propose the full-scale SSA (FSSSA), a population-based simulation algorithm for rule-based models. The FSSSA is implemented with sparse storage on populations of states to save extra computational effort. In numerical experiments, our new methods are compared with the SSA and the NFA based on two realistic biochemical models. Our results show that for models with complex network composed of multi-state species, the FSSSA can exhibit the best efficiency. This work has been published in [36, 37].

1.2 The Multi-Scale Feature

Besides the multi-state feature, another well-known challenge for stochastic simulation algorithms is related to the *multi-scale* feature of a system. In a complex biochemical system, different chemical reactions may reside in multiple scales. The challenge may first come from scale differences in reaction propensities. Fast reactions with larger propensities usually reach equilibrium very quickly. However, they are often "back-and-forth" reactions, which basically cancel each other's effect to the system dynamics. While the entire reaction firing simulation is dominated by these fast reactions, the impact of slower reactions is actually more important for the system behavior. It is often not necessary to spend most of the computational time simulating those fast but stable reactions. Meanwhile, the scale differences in numbers of reactant molecules is another source of the challenge. Reactions with large numbers of reactant molecules usually produce much less stochastic effects than other reactions. In this case, one may prefer to simulate them deterministically to avoid unnecessary computational burden.

In our research, two stochastic models of eukaryotic cell cycles exhibit interesting multi-scale features. The first one was developed by Kar et al. [11] (see Chapter 2), based on a simple cell cycle model by Tyson and Novak [38]. The multi-scale feature in this model is present in numbers of molecules between the protein level and the gene expression level. While numbers of molecules of proteins can reach several thousands, the gene and mRNA species usually have less than 10 molecules. It is fair to assume the existence of a scale difference

between reactions on the protein level and the gene expression level. To explore possible efficiency improvement from this scale difference, we first apply Haseltine and Rawlings' hybrid method [22] on Kar et al.'s cell cycle model. This method basically uses deterministic scheme for "slow" reactions and stochastic scheme for "fast" reactions. However, the existing strategies for partitioning the system lead to either large errors or low simulation efficiency. We propose a new partitioning strategy based on ideas from the tau-leaping method and the slow-scale SSA method. We apply different partitioning strategies and compare them with our new strategy in accuracy and efficiency. We demonstrate that, with our new strategy and properly selected parameters, Haseltine and Rawlings' hybrid method reproduces almost the same statistics as the original SSA method but saves significantly in the computational cost. Through this analysis, we also discover a simple but appropriate partitioning strategy for the cell cycle model: Treat all reactions related to gene expression as slow reactions and the rest as fast reactions. Biologically, it also matches with our intuition that most of the intrinsic noise arises at the gene expression level due to the low numbers of molecules of genes and mRNAs. On the other hand, this discovery implies a new approach to convert a deterministic cell cycle model into its stochastic counterpart. To build a new stochastic cell cycle model, we propose to keep the original ODEs of the proteins as the deterministic model. Meanwhile, stochastic elementary reactions related to gene expression are added as slow reactions and simulated with the SSA. Our new model has a much smaller system size and simpler system structure compared to Kar et al.'s model. Moreover, it does not require the inefficient full stochastic simulation. We present numerical results to demonstrate that

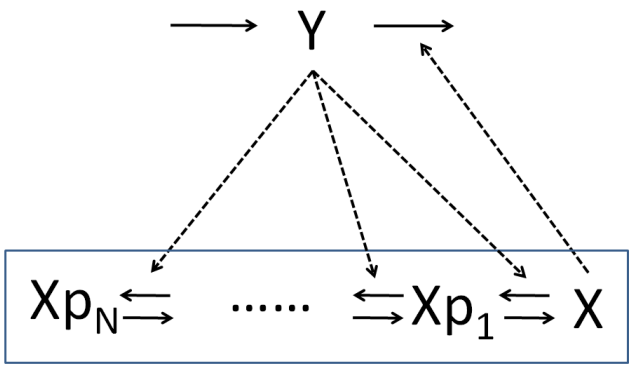


Figure 1.4: Bistable switch based on multiple site phosphorylation. Dashed arrows indicate enzyme activity.

the new model is able to correctly capture both the average dynamics and the stochastic noise generated inside the cell cycle. This work has been published in the Journal of Chemical Physics [39].

Another cell cycle model was developed by Barik et al. [12] (see Chapter 2). It is a stochastic model for the budding yeast cell cycle based on multiple site phosphorylations. To maintain correct cell behaviors, several important proteins are assumed to have multiple levels of phosphorylations. For simplicity, we focus on the core component of this model, the bistable switch, as shown in Figure 1.4.

Protein X is modeled with multiple site phosphorylation and thus is assumed to have up to N levels of phosphorylation. The (de-)phosphorylation reactions of X automatically form a reversible reaction chain. Due to this special setting, Barik et al.'s model actually possesses

both the multi-state and multi-scale features. We first discuss its multi-scale feature. In the bistable switch, the reaction chain contains numerous "back and forth" reactions. Therefore, we try to explore for the scale difference between the reaction chain and the other reactions by assuming the former has higher firing frequencies. To quantify such scale difference, we propose to approximate the relaxation time of the reaction chain, and compare it with the average firing time interval of the other reactions. Then we apply the QSSA and Haseltine and Rawlings' hybrid method on the bistable switch and analyze their performance. With numerical results we conclude that if the approximated relaxation time is much smaller than the average firing interval of the slow reactions, the QSSA can be accurate. Haseltine and Rawlings' hybrid method maintains accurate under a less strict condition. However it has lower computational efficiency than the QSSA. On the other hand, similar to Kar et al.'s model, Barik et al.'s model also contains mRNA species for most of the proteins. These mRNAs are assumed to be the major source of the noise. So in future work it should be a valid research direction for us to use Haseltine and Rawlings' hybrid method with the same partitioning strategy as we used for Kar et al.'s model. Now we discuss the multi-state feature of this model. In Barik et al.'s model, due to multiple site phosphorylation, most of its reactions are (de-)phosphorylation reactions and repeated binding and degradation reactions. The total number of reactions is over 400 and total number of species is around 90. This huge system size definitely presents great simulation challenge to the SSA. However, species modeled with multiple site phosphorylation automatically qualify as multi-state species. In this case, by using rule-based modeling, we reduce this model to one with 66 rules and 23

species. We apply our rule-based simulation methods (PNFA and FSSSA) on it and present numerical results to show that our methods have better efficiency than the SSA. This work has been published in [40].

Chapter 2

Literature Review

2.1 SSA

The SSA is an exact simulation scheme as it follows the same probability distribution as ruled by the chemical master equation (CME). The algorithm assumes that the systems are well-stirred and all the chemical reactions are elementary. Since the state variables in the SSA represent numbers of molecules, it is also a *population-based* method. Gillespie developed two equivalent formulations of the SSA: the Direct Method (DM) and the First Reaction Method (FRM) [13, 14]. The detailed implementation of DM is provided in Section 3.1.1. Later progress has been made to improve the implementation of the SSA. Gibson and Bruck proposed the Next Reaction Method (NRM) [26] which can be viewed as an extension of the FRM. The NRM saves computational cost as it generates much less random numbers than

the FRM. Cao et al. proposed the Optimized Direct Method (ODM) [27] which improves the efficiency of the DM by sorting propensity indices before simulation. The larger propensities are placed at the front of the search list so that the searching cost is minimized. McCollum et al. proposed the Sorting Direct Method (SDM) [41] which further improves the simulation efficiency of the ODM by using dynamically and loosely sorted propensity indices. As variations of the SSA, the ODM and SDM still are exact, population-based methods. Their performance is also problem-dependent.

2.2 Multi-State Simulation Methods

The StochSim was one of the earliest particle-based stochastic simulation algorithms designed to handle multi-state situations. It was developed by Morton-Firth and Bray [15, 16, 17] and successfully applied on models of bacterial chemotaxis. With its particle-based nature, the StochSim treats all reacting molecules as individual objects with their own properties, such as conformation states, velocity, spatial information and so on. Such a special feature makes StochSim extendable for handling stochastic simulation on multi-state variables, and its efficiency is not affected by the number of reactions present in the system [42]. The detailed implementation of the algorithm is provided in Section 3.1.2. Comparing StochSim with the SSA, Shimizu and Bray showed an equivalence of the physical assumptions of the two methods [17]. Another work by Pettigrew and Resat [43] compared them in terms of computational efficiency. However, this comparison was mostly based on a particular model.

Another way to deal with the multi-state feature is to use *rule-based modeling*, as proposed by Hlavacek et al. [34, 35]. The idea is to generalize a group of chemical reactions into one rule if all reactions in the group share the same binding properties of their reactants and products. The resulting rule-based models can be much smaller and simpler than conventional models that describe explicitly each individual chemical reaction. This can simplify both the conceptual complexity of the model for the modeler and of the simulation. Although models configured by the StochSim can also be easily converted to rule-based models, the rule-based modeling technique is expected to describe more generalized particle-based models than the StochSim. Based on a formal language specification for biochemical modeling, Danos et al. [44] developed a stochastic simulation method specifically for rule-based models. It modifies Gillespie's original SSA by substituting rules for reactions and using a particle-based simulation scheme. Later, Yang et al. [45] generalized this method so that a rule can have different rate constants for different states of reactants. Following their work, Sneddon et al. [18] developed a new rule-based model simulator called the Network-Free Stochastic Simulator (NFSim), which uses the methods presented in Danos' and Yang's work. We refer to this method as the Network-Free Algorithm (NFA). The detailed implementation of the NFA is provided in Section 5.1.2. The particle-based NFA is able to efficiently manage the multi-state situation, and it does not suffer from a fixed and underestimated step size as the StochSim does. On the other hand, the NFA creates objects for all molecules in the system, which may not always be desirable for single-state species with large populations in a system. Treating every molecule as an object results in both greater space requirements

and extra computational time in data structure maintenance.

2.3 Multi-Scale Simulation Methods

To deal with the multi-scale situation, several approximate simulation strategies have been proposed. One group of approximation methods tries to take advantage of the multi-scale characteristics observed in the reactant population: Some species are present in larger population numbers than the others. If all reactants have relatively large populations, one can represent a stochastic system by chemical Langevin equations (CLEs) and solve them as stochastic differential equations [46], or directly apply tau-leap methods [47, 48] which approximate the numbers of reactions by Poisson random numbers. The other group of approximation methods tries to take advantage of the multi-scale feature in the reactions: Some reactions occur much more frequently than others (these are called "fast" reactions, in contrast to "slow" reactions that occur comparatively infrequently). In the deterministic regime, quasi-steady state (QSS) and partial equilibrium (PE) assumptions are often used for approximation of simulation [49, 50, 51]. In the time scale of interest, the QSS assumption assumes that the changing rates of some species involved in the fast reactions are equal to zero, while the PE assumption assumes that the fast reactions are always in equilibrium. Both assumptions are used to avoid directly simulating the fast reactions. Instead, some algebraic equations are built and solved in each simulation step. The QSS assumption was then extended to the stochastic quasi-steady state assumption (SQSSA) [19], while the PE

assumption was extended to the slow-scale SSA (ssSSA) [20, 21], both for stochastic simulation of multi-scale biochemical systems. Although these studies are usually based on relatively strict assumptions and simple models, it was shown to be very useful in accelerating SSA simulations. In realistically large biochemical systems, species populations and reaction propensities can span several orders of magnitude. Thus it is not realistic to simulate a multi-scale system with only one method that works well in one scale. Instead, hybrid methods should be considered from a more practical, system-specific point of view. Cao et al. [52] proposed to partition the system based simply on the species population numbers. For species whose population numbers are less than a threshold, all related reactions are simulated by SSA, while other reactions are simulated by the tau-leaping method. Haseltine and Rawlings [22] proposed to partition a system into groups of slow and fast reactions. The partitioning criterion is determined by two thresholds set by the user before simulation. A reaction is put into the fast reaction group if its propensity is greater than the propensity threshold and the populations of all its reactants are greater than the population threshold. In this method, the fast reaction group is governed by ODEs or CLEs and the slow reaction group is simulated by Gillespie's direct method. A similar strategy was adopted by Salis et al. [53] with a small implementation difference, where fast reactions are approximated by CLEs and slow reactions are simulated by Gibson and Bruck's NRM.

2.4 Development of Stochastic Cell Cycle Models

The eukaryotic cell cycle is composed of the repetitive sequence of events that a cell grows, replicates its components and divides into two daughter cells. It is regulated by a complicated chemical reaction network. To model the cell cycle control system, theoretical biologists started with deterministic models using ordinary differential equations (ODEs) [38, 4]. Although deterministic cell cycle models can be precise and robust in many respects, experimental data exhibit considerable variability from cell to cell during cell growth and division [54]. Therefore, stochastic models and simulations are required to accurately model molecular fluctuations which are the major source of the noise in cell cycles. A natural way to build a stochastic model is to convert a deterministic model into its stochastic counterpart. One of the major difficulties in this type of conversion lies in the rate laws, which are often not elementary (mass-action) kinetics. The bistable switch always serves as the fundamental component of a cell cycle model [38]. The antagonism between two species (usually Cdh1 and Clb2) is established by two enzyme reactions with Michaelis-Menten rate laws. These phenomenological rate laws are required to generate certain amount of "nonlinearity" so that the bistable behavior can be observed. However, directly using phenomenological rate laws in the SSA may possibly generate incorrect stochastic results [55]. Therefore, a model based fully on mass-action kinetics for all reactions is desired for stochastic modeling. At the same time, the nonlinearity inside the switch has to be preserved.

To achieve this goal, Kar et al. [11] extended Tyson and Novak's model [38] by "unpacking"

the phenomenological deterministic reactions into a number of elementary stochastic reactions. In this process, mRNA variables (with low numbers of molecules) and other helper proteins are introduced into the network. The resulting network has a large size, but is able to model the repetitive cell cycle behavior on average and capture proper amounts of noise in the cell. More detailed background and development of this model is provided in Section 6.2.

On the other hand, another recent discovery by Qu et al. [56] shows that, modeling the regulative protein with multiple site phosphorylation is also likely to produce the "nonlinearity" in the system and the switch behavior. The idea is to assume the protein have multiple phosphorylation sites and levels, which enable it to undergo multiple (de-)phosphorylation reactions between different levels. With this idea, Barik et al. developed another stochastic model of the budding yeast cell cycle [12]. This model assumes 5 protein species have multiple site phosphorylation. To simulate proper amount of noise, mRNAs are also introduced for most of the proteins. This stochastic cell cycle model is able to generate accurate variability according to experimental data.

Part I

Multi-State Simulation Methods

Chapter 3

Comparison between SSA and StochSim

This chapter first introduces the SSA and the StochSim, and then gives a detailed comparison between the two methods in terms of accuracy and efficiency. Then we propose the Hybrid SSA (HSSA) to improve the performance of the original SSA in certain multi-state situations. Last, numerical experiments on the bacterial chemotaxis model and another sample model are presented.

3.1 Background

3.1.1 SSA

Suppose the system involves N molecular species $\{S_1, \ldots, S_N\}$. The state vector is denoted by $X(t) = (X_1(t), \ldots, X_N(t))$, where $X_i(t)$ is the number of molecules of species S_i at time t. M reaction channels $\{R_1, \ldots, R_M\}$ are involved in the system. A reaction channel is any chemical event that causes changes to the state variables. Assume that the system is well stirred and in thermal equilibrium. The dynamics of reaction channel R_j is characterized by the *propensity function* a_j and the *state change vector* $\nu_j = (\nu_{1j}, \ldots, \nu_{N,j})$: $a_j(x)dt$ gives the probability that one R_j reaction will occur in the next infinitesimal time interval $[t, t + dt)$, and ν_{ij} gives the change in the S_i molecule population induced by one R_j reaction.

The dynamics of the system can be simulated by the SSA [13, 14]. With $X(t) = x$, let $a_0(x) = \sum_{j=1}^{M} a_j(x)$. On each step, the SSA generates two random numbers r_1 and r_2 in $U(0, 1)$, the uniform distribution on the interval $(0, 1)$. The time for the next reaction to occur is given by $t + \tau$, where τ is given by

$$\tau = \frac{1}{a_0(x)} \log\left(\frac{1}{r_1}\right). \tag{3.1}$$

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

$$\sum_{l=1}^{j} a_l(x) > r_2 a_0(x). \tag{3.2}$$

The system states are updated by $X(t + \tau) = x + \nu_j$. The simulation proceeds to the next

occurring time, until it reaches the final time.

3.1.2 StochSim

Initialization

Before simulation, as a particle-based algorithm, the StochSim creates objects for all molecules with their own properties according to initial conditions, along with a computed number of *pseudo-molecule objects*. A look-up table is then constructed to describe the reaction possibilities for all reaction channels. The rows of the table list the first reactant and the columns list the second reactant (if the second reactant is a pseudo-molecule, it represents a uni-molecule reaction). The corresponding entry in the table shows the probability for the two molecules to have a reaction. If there are multiple reaction channels between the two molecules, this entry saves the sum of the probabilities of all involved reaction channels.

In order to distinguish uni-molecule reactions from bi-molecule reactions, pseudo-molecule objects are introduced with a fixed population $N_{psm:obj}$. $N_{psm:obj}$ is calculated so that the maximum possibility of the uni-molecule reactions is equal to that of the bi-molecule reactions. Let $k_{1,max}$ be the maximum reaction rate of all uni-molecule reactions, and $k_{2,max}$ be the maximum reaction rate of all bi-molecule reactions. Therefore,

$$N_{psm:obj} = Round(2 \cdot N_A V \cdot \frac{k_{1,max}}{k_{2,max}}), \tag{3.3}$$

where $Round(n)$ represents the positive integer nearest to n, N_A is the Avogadro constant, and V is the volume of the system.

Then, probabilities in the look-up table are calculated with the following formula. For a uni-molecule reaction,

$$p_{1,j} = \frac{k_{1,j} N_{m:obj} (N_{m:obj} + N_{psm:obj} - 1) \Delta t}{N_{psm:obj}}, \tag{3.4}$$

and for a bi-molecule reaction,

$$p_{2,j} = \frac{k_{2,j} N_{m:obj} (N_{m:obj} + N_{psm:obj} - 1) \Delta t}{2 N_A V}, \tag{3.5}$$

where $N_{m:obj}$ is the total number of real molecule objects (different from pseudo-molecule objects) in the system, $k_{1,j}$ and $k_{2,j}$ are respectively the uni-molecule and bi-molecule rate constants for reaction j, and Δt is the time step size which is pre-determined from user's input.

In the last step before simulation, the time step size Δt is optimized. The criteria is that the maximum probability in the look-up table must be smaller than a constant $MAXP$. In the implementation of the StochSim [57], $MAXP$ is set to 0.599, and the corresponding formula for Δt is given by

$$\Delta t = \frac{MAXP}{k_{1,max} N_{m:obj} + \frac{k_{2,max} N_{m:obj} (N_{m:obj} - 1)}{2 N_A V}}. \tag{3.6}$$

This optimized Δt then replaces the previous value, and is used to recalculate all the probabilities in the look-up table.

Note that $N_{m:obj}$ is a constant and limits the application of the algorithm. For example, in a system with a binding reaction

$$A + B \longrightarrow C, \tag{3.7}$$

the total number of molecules will decrease by 1 after firing this reaction. In the implementation [57] of the StochSim, $N_{m:obj}$ is chosen as the maximum possible total number of real molecule objects, equal to or larger than the actual total population. Therefore, certain molecules of dummy species are introduced to compensate the change of total number of molecules. So the reaction (3.7) can be read as

$$A + B \longrightarrow C + Dummy. \tag{3.8}$$

With this simplification, the algorithm keeps $N_{m:obj}$ a fixed number.

Simulation

The whole simulation time interval is evenly divided into a series of discrete time steps of a fixed size Δt. In each time step, the StochSim proceeds with the following steps.

1. Randomly select the first reactant from all real molecule objects using uniform distri-

bution.

2. Randomly select the second reactant from all real molecule objects and pseudo-molecule objects using uniform distribution.

3. Search the look-up table for possible reactions between the two selected objects. If no corresponding entry is found, the StochSim concludes that no reaction occurs in this time step. Otherwise a uniform random number in [0, 1) is generated and compared with the probability retrieved from the table. If the random number is larger, the StochSim concludes that no reaction occurs in this time step. Otherwise there is a reaction between these two molecules. If there is only one possible reaction channel between these two molecules, the reaction is selected to fire. Otherwise, the code selects one of the possible reaction channels in a similar way as the equation (3.2) in the SSA.

4. Update the system accordingly and proceed the simulation to the next time step.

3.2 Accuracy Analysis on StochSim

The implementation details of the SSA and StochSim are quite different. However, it was stated in Shimizu and Bray's book [17] that these two methods are based on equivalent fundamental physics assumptions. Pettigrew and Resat [43] also showed that these two methods generated similar distribution plots for a particular model. In this section, we analyze the StochSim from a different angle. Since the SSA is an exact stochastic simulation

scheme, it is natural to first examine the accuracy of the StochSim.

Instead of directly comparing it with the SSA, we first compare the SSA with a simple simulation procedure. Suppose that we equally divide the time interval into many small time slices Δt. Because the probability that the R_j reaction channel will fire in the next infinitesimal time interval $[t, t + dt)$ is given by $a_j(x)dt$, when Δt is sufficiently small the probability that one R_j reaction will occur in the time interval $[t, t+\Delta t)$ can be *approximated* by $a_j(x)\Delta t$. We can then implement the following simulation procedure.

Simulation Procedure 3.2.1 *Suppose at time t the system is at $X(t) = x$ and the probability that one R_j reaction will occur in the time interval $[t, t + \Delta t)$ is given by $a_j(x)\Delta t$. In each Δt time slice, a random number r is generated uniformly on interval $[0,\ 1)$ and compared with $a_0(x)\Delta t$. If $a_0(x)\Delta t > r$, one reaction will fire in this small time slice Δt and the reaction channel index j can be selected the same as (3.2) in the standard SSA procedure. We let the time proceed to $t + \Delta t$ and update the state variable by $X(t + \Delta t) = x + \nu_j$. Otherwise, no reaction will fire in this time interval. We just let the time proceed to $t + \Delta t$ with no state variable update.*

One can easily see that this simulation procedure is not exact. If Δt is chosen larger than $\frac{1}{a_0(x)}$, the *probability* $a_0(x)\Delta t$ will be greater than 1, and that is not allowed. Will this procedure be exact if $\Delta t < \frac{1}{a_0(x)}$? No. Note that simulation procedure 3.2.1 implies that $1 - a_0(x)\Delta t$ is the probability that no reaction will occur in the next time interval Δt. However, one can easily derive that the probability that no reaction will fire in the next Δt

for any $\Delta t > 0$ is

$$e^{-a_0(x)\Delta t} = 1 - a_0(x)\Delta t + \frac{1}{2}\left(a_0(x)\Delta t\right)^2 + O((\Delta t)^3). \tag{3.9}$$

We can see that $1 - a_0(x)\Delta t$ is the first-order approximation of (3.9) and the leading term for the error is given by $\frac{1}{2}\left[a_0(x)\Delta t\right]^2$. The leading term shows that if $a_0(x)\Delta t \leq 0.1$, which gives

$$\Delta t \leq \frac{0.1}{a_0(x)}, \tag{3.10}$$

the error of the first-order approximation can be estimated by 0.005. This Δt may be small enough so that the first-order approximation is acceptable and the histogram generated from simulation procedure 3.2.1 will be close to that generated by the SSA. However, if Δt has to be one magnitude smaller than $\frac{1}{a_0(x)}$, the chance that no reaction will fire in the next time step $[t,\ t + \Delta t)$ is relatively high. Thus before one reaction really fires in simulation procedure 3.2.1, there will be several (around 10) steps that no reaction fires at all. This situation can be illustrated in Figure 3.1.

Next we compare simulation procedure 3.2.1 and the StochSim. Let the fixed time steps in the StochSim and procedure 3.2.1 both be Δt. We have the following theorem.

Theorem 3.2.1 *The StochSim uses the possibility of $a_j(x)\delta t$ to describe one R_j reaction to occur in the time interval $[t, t+\delta t$. This possibility is as same as the approximated possibility*

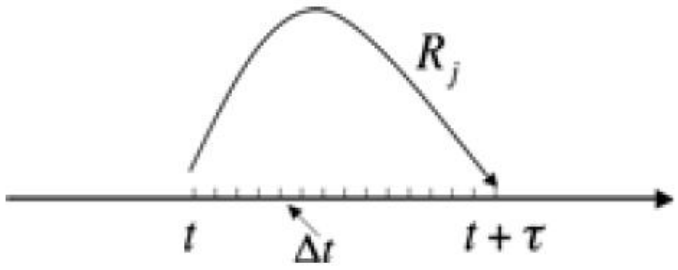


Figure 3.1: The time step comparison between the SSA and simulation procedure 3.2.1. To ensure the accuracy, for each reaction corresponding to one step size τ in the SSA, there must be several (around 10) steps in the time interval Δt where no reaction fires in simulation procedure 3.2.1.

in simulation procedure 3.2.1, when δt is sufficiently small.

PROOF. In the StochSim, two objects are randomly selected in the first two steps. The probability that these two could have a reaction is given by (3.4) or (3.5) if Δt is small.

Let us first look at a mono-molecule reaction R_j with a reactant S_i. In the assumption of simulation procedure 3.2.1, the probability that one R_j reaction will occur in the next time interval $[t, t+\Delta t)$ is given by $a_j(x)\Delta t$, where $a_j(x) = c_j x_i$. On the other hand, in the StochSim process R_j is selected when a S_i object is selected in the first step and a pseudo molecular object is selected in the second step. The probability that one S_i molecule is selected in the first step is $\frac{x_i}{N}$. The probability that one pseudo molecule is selected in the second step is $\frac{N_{psm:obj}}{N_{m:obj} + N_{psm:obj} - 1}$. Multiply them with the equation (3.4). The probability that an R_j reaction will fire in the next time interval $[t,\ t+\Delta t)$ in the StochSim is thus given by

$$\frac{x_i}{N_{m:obj}} \cdot \frac{N_{psm:obj}}{N_{m:obj} + N_{psm:obj} - 1} \cdot \frac{k_{1,j} N_{m:obj}(N_{m:obj} + N_{psm:obj} - 1)\Delta t}{N_{psm:obj}} = k_{1,j} x_i \Delta t. \quad (3.11)$$

Note that $c_j = k_{1,j}$ for a uni-molecule reaction [13]. Thus the StochSim and Procedure 3.2.1 follow the same probability for a uni-molecule reaction.

For a bi-molecule reaction R_j between reactants S_i and S_k, Procedure 3.2.1 assumes that the probability that one R_j reaction will fire in the time interval $[t, t + \Delta t)$ is given by $a_j(x)\Delta t$, where $a_j(x) = c_j x_i x_k$. In the StochSim, R_j may fire only if S_i and S_k are selected in the first two steps. The probability that one S_i molecule is selected in the first step is $\frac{x_i}{N_{m:obj}}$, while the probability that one S_k molecule is selected in the second step is $\frac{x_k}{N_{m:obj} + N_{psm:obj} - 1}$. The probability that S_k is selected in the first step and S_i is selected in the second step is the same. Thus the probability that one S_i molecule and one S_k molecule are selected in the first two steps is $\frac{2x_i x_k}{N(N + N_0 - 1)}$. Multiply it with the equation (3.5). The probability that an R_j reaction will fire in the next time interval Δt in the StochSim is given by

$$\frac{2x_i x_k}{N_{m:obj}(N_{m:obj} + N_{psm:obj} - 1)} \cdot \frac{k_{2,j} N_{m:obj}(N_{m:obj} + N_{psm:obj} - 1)\Delta t}{2N_A V} = \frac{k_{2,j}}{N_A V} x_i x_k \Delta t. \quad (3.12)$$

Note that $c_j = \frac{k_{2,j}}{N_A V}$ for a bi-molecule reaction between two different species [13]. Again we see that the StochSim and Procedure 3.2.1 follow the same probability.

For a bi-molecule reaction R_j between two S_i molecules, Procedure 3.2.1 assumes that the probability that one R_j reaction will fire in the time interval $[t, t + \Delta t)$ is given by $a_j(x)\Delta t$,

where $a_j(x) = \frac{1}{2}c_j x_i(x_i - 1)$. In the StochSim, R_j may fire only if S_i is selected in both steps. The probability that one S_i molecule is selected in the first step is $\frac{x_i}{N_{m:obj}}$, while the probability that a different S_i molecule is selected in the second step is $\frac{x_i - 1}{N_{m:obj} + N_{psm:obj} - 1}$. Similar to the case of bi-molecule reaction between two different species, these two molecules can be selected with different order. Thus the probability that two S_i molecules are selected in the first two steps is $\frac{2x_i(x_i - 1)}{N_{m:obj}(N_{m:obj} + N_{PSm:obj} - 1)}$. Multiply it with the equation (3.5). The probability that an R_j reaction will fire in the next time interval Δt in the StochSim is given by

$$\frac{2x_i(x_i - 1)}{N_{m:obj}(N_{m:obj} + N_{psm:obj} - 1)} \cdot \frac{k_{2,j}N_{m:obj}(N_{m:obj} + N_{psm:obj} - 1)\Delta t}{2N_A V} = \frac{k_{2,j}}{N_A V}x_i(x_i - 1)\Delta t. \tag{3.13}$$

Note that $c_j = \frac{2k_{2,j}}{N_A V}$ [1] for a bi-molecule reaction between the same species [13]. Again we see that the StochSim and Procedure 3.2.1 follow the same probability. Thus we have the theorem. □

From this theorem we can see that when Δt is sufficiently small, the StochSim follows the same probabilities as in Procedure 3.2.1, so it can also be viewed as a first-order approximation to the exact SSA.

We can see this point through analysis on the following example.

Example 1: Suppose a simple system has only one species A and one decay reaction

[1] The relation between c_j and $k_{2,j}$ in this case is different from the case of one bi-molecule reaction between two different species, as pointed out in [13].

$$A \xrightarrow{k} X, \tag{3.14}$$

where the population of A remains constant, and k is the reaction rate constant. We let the system proceed by duration of T and count the number of the firings of the only reaction (3.14), denoted by n_f. One can easily see that the possibility for the reaction to fire in the next infinitesimal time $[t, t+dt)$ is given by $|A|kdt$ where $|A|$ is the population of species A, and that the whole process is a Poisson process. Obviously, the theoretical mean value and variance of n_f are both $|A|kT$. Then we use the StochSim to simulate this system. Since this system only has a uni-molecule reaction, $N_{psm:obj}$ is set to ∞ [57]. Therefore according to (3.6), Δt is given by

$$\Delta t = \frac{MAXP}{k|A|}, \tag{3.15}$$

and according to (3.4) and (3.15), the possibility in the look-up table is given by

$$p_1 = k|A|\Delta t = MAXP. \tag{3.16}$$

Note that the StochSim evenly divides T into several time steps and in all steps the probability for the reaction to fire is the same, so n_f follows a binomial distribution $B(n, p)$, where

$$n = \frac{T}{\Delta t} = \frac{Tk|A|}{MAXP} \tag{3.17}$$

and

$$p = p_1 = MAXP. \tag{3.18}$$

Therefore with simulation of the StochSim, the mean value and variance of n_f are respectively $np = |A|kT$ and $np(1-p) = |A|kT(1-MAXP)$. Although the StochSim generates the correct mean value, the variance is dependent on $MAXP$. Only when $MAXP$ is very small compared to 1.0, such as 0.1, so that Δt at least satisfies (3.10), the distribution of n_f can be approximately accurate. However, the StochSim's real implementation value 0.599 for $MAXP$ will apparently make the simulation on this model deviate from the right distribution. Again, in order to obtain the accurate distribution, the StochSim has to set Δt sufficiently small.

3.3 Efficiency Comparison

Base on the previous analysis, we are enabled to compare the efficiency of the SSA and the StochSim. The computational cost of a dynamic simulation algorithm is composed of two parts: the average computational cost for each time step and the total number of time steps in a simulation. The computational costs for the SSA and simulation procedure 3.2.1 in each step are both $O(M)$. But there are much more steps in Procedure 3.2.1. Thus we

can conclude that simulation procedure 3.2.1 is an approximation to the SSA with higher computational cost. Obviously it is not a good strategy in stochastic simulation.

Now between Procedure 3.2.1 and the StochSim, since their computational costs are proportional to $\frac{1}{\Delta t}$, it is preferred to have a large Δt. However, both are accurate only when Δt is selected sufficiently small. This is a shared restriction on Δt enforced on both procedures. Moreover, the StochSim procedure requires all $p_{1,j}$'s in (3.4) and $p_{2,j}$'s in (3.5) be not larger than *MAXP*, a constant not larger than 1.0. This extra restriction on Δt can be even tighter than the first one.

To see this point, let us consider the following example.

Example 2: Suppose there are three species S_1, S_2 and S_3 but only one reaction channel R_1 between S_1 and S_2.

$$S_1 + S_2 \xrightarrow{c} S_3 + S_2. \tag{3.19}$$

Let $X_1 = M_0$, $X_2 = 1$ and $X_3 = 0$, where $M_0 >> 1$. Let $c = 1$. Thus the propensity function for R_1 is $a_1(x) = x_1 x_2$. According to the SSA, the mean time for the next R_1 reaction to fire is given by

$$\tau_{SSA} = \frac{1}{a_1(x)} = \frac{1}{M_0}. \tag{3.20}$$

In simulation procedure 3.2.1, the corresponding simulation time step Δt can be given by

$$\Delta t < \frac{0.1}{a_1(x)} = \frac{1}{10M_0}. \tag{3.21}$$

For the StochSim, with the maximal value $MAXP = 1.0$ and (3.5), we have

$$p_2 = \frac{k_2 N_{m:obj}(N_{m:obj} + N_{psm:obj})\Delta t}{2N_A V} < 1.$$

Here we already know that $c = \frac{k_2}{N_A V} = 1$. Thus the StochSim should satisfy the restriction

$$\frac{N_{m:obj}(N_{m:obj} + N_{psm:obj})\Delta t}{2} < 1, \tag{3.22}$$

which gives

$$\Delta t < \frac{2}{N_{m:obj}(N_{m:obj} + N_{psm:obj})}. \tag{3.23}$$

Here $N_{m:obj}$ is the total number of molecules. $N_{m:obj} = M_0 + 1$. $N_{psm:obj}$ is the total number of pseudo-molecules. Since there is no mono-molecule reaction, $N_{psm:obj} = 0$. Then $\Delta t < \frac{2}{(M_0+1)^2}$. When M_0 is large, this restriction on Δt is much tighter than (3.21). This example demonstrates a situation where the StochSim selects a much smaller Δt than the one in Procedure 3.2.1 although Theorem 3.2.1 states that they both follow the same probability when Δt is sufficiently small.

Let $M_0 = 10^4$, equations (3.20) and (3.23) imply that for this example the number of steps the StochSim needs is about $5,000$ times as what the SSA needs. The efficiency is very

low. Of course this example is an extreme case. According to (3.6), a too small step size in the StochSim often arises in situations where some reaction rates related to the species with large populations are relatively small, or there are reactant species with relatively small populations compared to the total populations $N_{m:obj}$. In these cases, the time step in the StochSim will be much smaller than the mean time step in Procedure 3.2.1 and the SSA, and cause very low efficiency.

According to the above analysis, we can conclude that the StochSim can usually take a much larger number of time steps than Procedure 3.2.1, which is still about one magnitude less efficient than the SSA. On the other hand, we should also note that, the computational cost in a single step for StochSim is $O(1)$ while for the SSA it is $O(M)$. If the average step sizes of the SSA and the StochSim are very close to each other and $M \gg 1$, the StochSim can exhibit higher efficiency. However, the efficiency difference in a single step is usually dominated by the difference in step sizes, especially for multi-scale cases such as Example 2. In this case, the efficiency of the StochSim can be much lower than that of the SSA.

3.4 Multi-State Situation

3.4.1 Multi-state species

Although for a multi-scale system where there can be magnitude difference between populations or reaction rates, the efficiency of the StochSim is much lower than that of the SSA. The

StochSim could have advantages for some systems if its computational cost in a single step is much lower than that of the SSA. A typical situation is when multi-state species are involved. Multi-state species often appear in biological systems where one molecule may change its characteristics depending on changes on its many binding sites, such as phosphorylation or methylation. If a molecule has 10 binding sites, depending on the states of all these binding sites, one molecule may exhibit $2^{10} = 1024$ different states. When it has 20 binding sites, the number of possible states rises to a million. If multi-state species can combine in many different ways, the combinatorial complexity may lead to a very large system size [34, 44, 45]. In the structure of the traditional SSA simulation, each possible state should be assigned with an independent state variable. If each of them may react with other species in the system, N, the number of species, and M, the number of reaction channels, may become dramatically large. Note that the computational cost of the SSA in a single step is $O(M)$. If a multi-state species is involved with many reaction channels, we have $M = O(N_{states})$, where N_{states} denotes the number of possible states for the multi-state species. N_{states} can be very large, which results in a large M. The computational cost of the SSA could be very high in this case. However, since the StochSim treats molecules as individual objects, it need not introduce a large number of species and reaction channels. Its computational cost for a single step does not change with N_{states}. Thus when N_{states} is large, the computational cost in each step of the StochSim can be much smaller than that of the SSA, which may compensate the extra cost in the total number of steps as analyzed in the previous section. In that case, the StochSim shows advantages over the SSA.

Example 3: Consider a system with three types of species X, Y and E, where E is an enzyme with $1,000$ different states (10 binding sites). Each state has different characteristics for its enzyme ability. And the population of E is small. Three types of reactions are considered.

$$X \xrightarrow{E_n} Y, \tag{3.24}$$

$$E_n \longrightarrow E_{n+1}, \tag{3.25}$$

$$E_{n+1} \longrightarrow E_n, \tag{3.26}$$

where E_n represents E in state n. For the SSA to simulate this system, because each state of E is formulated as an individual species and each of the three reaction types (3.24-3.26) will be correspondingly extended to around 1,000 reaction channels, we have $N = 1,002$ and $M \approx 3,000$. However, the StochSim can group all these reaction channels into just three channels (3.24-3.26) by treating the 1,000 reaction channels as one. When an E molecule is picked in the first two steps of the StochSim, it must have been in a particular state. When one reaction fires and this molecule changes its state, the StochSim procedure only needs to change the corresponding state for this molecule.

3.4.2 The hybrid SSA

For many biological systems, the populations of species are of multi-scale. Most species are present with large or moderate populations and do not have the multi-state problem as

discussed in Section 3.4.1. There are a few species with multi-state characteristics. If the multi-state species present with small populations, it is more efficient to treat each multi-state molecule as an independent object. We call this strategy the hybrid SSA (HSSA). The simulation procedure of the HSSA is very similar to that of the standard SSA. The difference lies in the classification of species and reaction channels. A system contains two types of species: normal species and multi-state species. In the HSSA, normal species remain the same as in the standard SSA while each molecule of the multi-state species is stored as an indexed object. Then this system contains three types of indexed reactions.

1. Reactions among normal species. This type remains the same as in the standard SSA.

2. Reactions involved with only one multi-state object. They include uni-molecule reactions of a multi-state object and bi-molecule reactions between a multi-state object and a non-multi-state species.

3. Reactions between two multi-state objects.

In the simulation, the total propensity function $a_0(x)$ is the sum of all reactions. The time step τ and the reaction index j are calculated using (3.1) and (3.2). If the index j points to the first type, the system is updated as in the standard SSA. Otherwise, the firing of a reaction will cause a state change of one or two multi-state objects. Each involved object changes its corresponding state and updates its rate constants. When a new multi-state molecule is produced, a new object is created into the system. When an existing multi-state molecule degrades, the corresponding object is eliminated from the system. Thus the

numbers of species and reaction channels vary. For certain systems, the dynamical changes in the numbers of species and reaction channels may be frequent. Such examples can be found in [58] where the chemical network is updated dynamically and the standard SSA is applied to the dynamically varying chemical network. If the multi-state species are generated and eliminated with small populations, the HSSA strategy will show good efficiency.

Let us consider Example 2 again. Assume that the total population of E is E_T. As discussed before, the standard SSA will have 1002 species and about 3000 reaction channels. However, if we apply the HSSA method, each E molecule is treated as a multi-state object. For each E object the three reaction channels (3.24-3.26) have local copies. Thus $N = 2 + E_T$ and $M = 3E_T$. When E_T is small, the efficiency gain is great. When E_T increases, the efficiency gain decreases. When E_T reaches around $1,000$, it will be less efficient than the standard SSA.

3.5 Numerical Experiments

In this section we present numerical experiments for the comparison of the SSA and the StochSim.

3.5.1 Bacteria chemotaxis model

The first example we use is the bacteria chemotaxis model on which the StochSim has been applied successfully. This model contains 12 reaction channels with the assumption that some fast reaction channels always remain in equilibrium [15]. The enzyme TTWWAA plays an important role in this model. It has multiple states and can participate in various types of reactions such as phosphorylation, methylation, and binding with other chemicals. The StochSim models each molecule of the enzyme TTWWAA as an object with many different states. To implement the SSA, we have to transform different states of the enzyme TTWWAA into different species. Each species corresponds to one state of the enzyme. By doing so, we increase the number of reactants and reactions in the system. The resulted model contains 28 reaction channels.

Figure 3.2 shows the trajectories of the active enzyme TTWWAA generated by the aaaaaaSSA and the StochSim respectively. We see that the trajectories match well with each other. The difference is mostly due to the fluctuation inherent with the stochastic simulation.

We listed the means and variances of the total population of the active TTWWAA in Table 3.1. The corresponding histograms are shown in Figure 3.3. We can see that the results from the StochSim and the SSA are very close. But the average tau value of the SSA is 217 times larger than the Δt for StochSim, while the simulation time of the SSA is 31 times faster than that of StochSim. Note that here for the fair comparison, the StochSim has been rewritten in C language to improve the efficiency. For the original StochSim package,

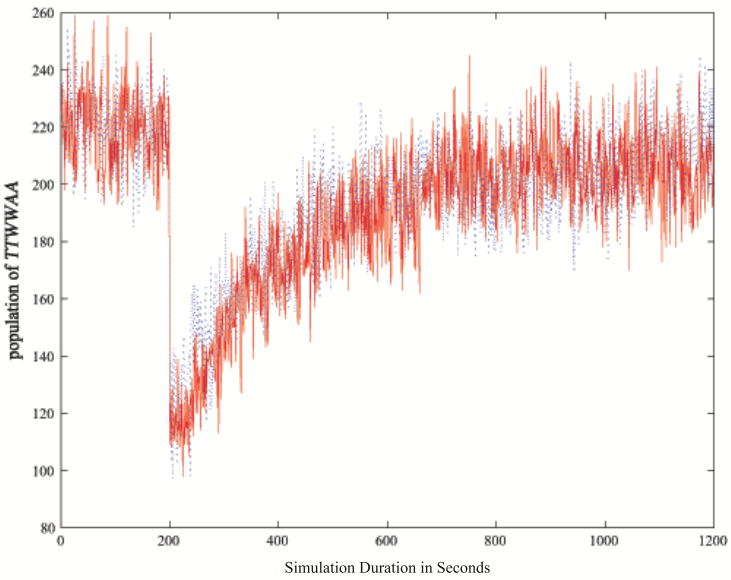


Figure 3.2: Trajectory comparison between SSA and StochSim. The solid red trajectory is simulated by the SSA, and the dotted blue one is by the StochSim. At the time of 200s, there is a sudden drop because a certain amount of aspartate is added into the system, repressing the activation of TTWWAA.

$10,000$ simulations took $219,280$ seconds CPU time, which is 5.6 times slower than our simple implementation in C language.

	Mean	Variance	Simulation time	Average Stepsize
StochSim	210.82	202.83	39,202s	3.6×10^{-5}
SSA	210.12	207.82	1,268s	7.9×10^{-3}

Table 3.1: Numerical results on chemotaxis model by the SSA and StochSim. Comparison of the means, variances, CPU times and average step sizes by the SSA and the StochSim on the chemotaxis model for 10,000 runs.

3.5.2 Efficiency comparison on a simple example

We implemented the StochSim, the SSA and the HSSA methods on a modified version of Example 3. The reactions are listed below

$$\begin{aligned} E_n + X &\xrightarrow{k_1} E_n + Y, \\ Y &\xrightarrow{k_2} X, \\ E_n &\xrightarrow{k_3} E_{n+1}, \\ E_n &\xrightarrow{k_4} E_{n-1}, \end{aligned} \tag{3.27}$$

where E_n represents the state n in species E and $1 \leq n \leq 1,000$. The reaction rate $k_1 = 5 \times 10^3 n$ [2], where n is the index of the corresponding state of E. $k_2 = 1$ and $k_3 = k_4 = 0.1$. First we fix the initial population of $X = 100$ and $Y = 0$ and increase E_T, the population of

[2] k_1 is much larger than other reaction rates because it is a bi-molecule reaction rate.

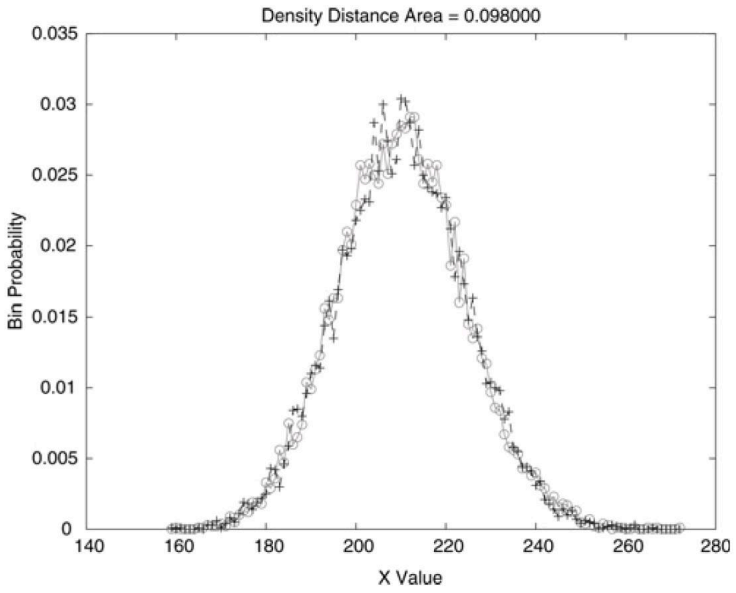


Figure 3.3: Distribution comparison between SSA and StochSim. The histograms of the active TTWWAA are simulated by the SSA (cross) and StochSim (circle).

E, from 1 to $10,000$.

The CPU time for different methods are listed in Table 3.2. For this simple example, when the population of E is small, the HSSA is the most efficient, and the StochSim is a little more efficient than the SSA. As E_T increases by a factor of 10, the CPU time for the SSA increases with a factor between 6 and 10. For the StochSim, when $E_T < 100$, the CPU time increased little. After $E_T > 100$, the CPU time shows super-linear increase. The CPU time of the HSSA shows even more obvious super-linear increase trend. As E_T becomes large, the HSSA becomes the slowest method.

The simulation results can be explained with complexity analysis. The CPU time is decided by the computational cost in a single step multiplied by the total number of steps. For

Population of E	1	10	100	1,000	10,000
SSA	0.81	6.93	58.29	324	2,013
StochSim	0.31	0.34	0.90	11.42	666
HSSA	0.013	0.11	6.64	425	22,122

Table 3.2: CPU time comparison among SSA, StochSim and HSSA. Comparison of CPU times in seconds among the SSA, the StochSim, and the HSSA for 1,000 runs of the model (3.27)

the SSA, as E_T increases, the propensity values $a_j(x) = O(E_T)$ except the second reaction channel in (3.27). The computational cost in a single step of the SSA does not change with E_T. The number of steps is of order $O(\frac{1}{\tau})$. Because the τ value is of order $O(\frac{1}{a_0})$, the number of total simulation steps is then of order $O(a_0)$. But for this example, $a_0(x) = \sum_j a_j(x)$ is of order $O(E_T)$. Thus the CPU time is of order $O(E_T)$. For the StochSim, when E_T increases, the computational cost in a single step does not change much. The CPU time is proportional to $\frac{1}{\Delta t}$. From (3.6) we know that the CPU time is of order $O(N_{m:obj}^2)$. In this example, $N_{m:obj} = 100 + E_T$. When $E_T < 100$, $N_{m:obj}$ does not change much as E_T increases. Thus the CPU time does not increase much. But when $E_T > 100$, $N_{m:obj}$ increases linearly with E_T. Thus the CPU time increases quadratically with E_T. For the HSSA, since it is still the SSA method, the number of steps should be the same as the situation in the standard SSA. Thus the number of the steps for the HSSA is in order $O(E_T)$. However, its computational cost in a single step is $O(M)$. When E_T is small, M is small. This cost is small. When E_T increases, each new object will have its own copy of reaction channels. We have $M = O(E_T)$. Thus the computational cost in a single step is $O(E_T)$. Multiplying the computational cost

in a single step and the total number of steps, we know that the CPU time for the HSSA is of order $O(E_T^2)$.

From this example we can see that the HSSA works well when E_T is small. When the total population increases, the efficiency of the StochSim drops quadratically. But with a large E_T, the StochSim still shows advantages for systems involved with multi-state species. Further improvement of HSSA is in need to combine the advantages of the StochSim and the SSA.

Chapter 4

A Hybrid Strategy Combining Improved StochSim and SSA

The HSSA presented in the previous chapter shows a high efficiency when the number of multi-state molecules is small. However, when the number of multi-state molecules increases, the large numbers of species and reaction channels still present a challenge. A good solution is still in demand. In this chapter, we first improve the original StochSim for enhanced simulation efficiency. Then, we propose a hybrid scheme that effectively combines the advantages of the StochSim and the SSA. Then results of numerical experiments are provided.

4.1 Motivation

From the comparison between the SSA and the StochSim we know the following results: the SSA has the advantage in simulating systems with large populations, while StochSim has the capability to overcome the multi-state obstacle [43]. However, often we have complicated models with high populations of different species of molecules, at the same time containing one or more species of multi-state complex molecules. In this case, we need a hybrid scheme to combine the advantages of both the SSA and the StochSim and overcome each algorithm's drawback. This hybrid scheme should exhibit higher efficiency than either of the SSA and the StochSim. To achieve this goal, we must notice that the major difference between the two methods is that the StochSim takes a fixed and very small step size while the SSA selects an adaptive one in each step. As a result, the combination of the two requires the StochSim use a new strategy to generate its step size.

4.1.1 An improved StochSim

To facilitate the combination of the SSA and StochSim, we propose an improved version of the StochSim with modifications in two aspects to avoid generating a fixed and too small step size.

Adaptive step size

In the StochSim, the total number of molecules $N_{m:obj}$ is maximized and fixed. The step size Δt is also fixed before simulation by (3.6), and approximately proportional to the inverse of $N^2_{m:obj}$. For many realistic systems the actual total number of molecules in system may change dynamically. Taking the maximal value for the number of molecule objects can result in a very small fixed Δt, thus a low efficiency. Therefore, it is desirable to have an adaptive step size Δt, varying with the actual total number of molecules in the system.

In our improved version of the StochSim, for those models whose total number of molecules may change drastically over time, we modify the algorithm as follows. First, no look-up table is needed any more. In the simulation, at each time step, use $N_{m:obj}$ as the actual total number of molecules in the system and calculate Δt using (3.6). Then when molecules are selected, the probability for the corresponding reaction is calculated using (3.4) or (3.5).

Eliminating pseudo-molecule

The original StochSim uses pseudo-molecule objects to differentiate uni-molecule reactions from bi-molecule reactions. Although having this definition does facilitate deriving the probability formula, it adds redundant information into the system. We have derived an improved algorithm that does not need pseudo-molecules. We will show that this change makes the method more efficient.

To avoid pseudo-molecules, we further modify the StochSim based on the improved version

described in the previous section. In initialization, we no longer calculate $N_{psm:obj}$ or create any pseudo-molecule. In each time step, we use a different formula to optimize Δt, and different formula to calculate the probability for the possible reaction. The whole simulation procedure is given as the following:

1. Calculate the actual number of molecules in system. Calculate adaptive Δt with (4.3).

2. Randomly select the first reactant from all molecules, using uniform distribution.

3. Search for possible uni-molecule reactions for the selected molecule. If no corresponding reaction is found, go to step 4. Otherwise, calculate the corresponding possibility using (4.1) and compare it with a uniform random number generated in $[0,\ 1)$. If the random number is larger, go to step 4. Otherwise there is a uni-molecule reaction to fire. If there is only one possible reaction channel for the selected molecule, select this reaction and go to step 6. Otherwise, the code selects one of the possible reaction channels in a similar way as (3.2) in the SSA and go to step 6.

4. Randomly select the second reactant from all molecules, using uniform distribution.

5. Search possible bi-molecule reactions between the two selected objects. If no corresponding reaction is found, the StochSim concludes that no reaction occurs in this time step. Otherwise, calculate the corresponding possibility using (4.2) and compare it with a uniform random number generated in $[0,\ 1)$. If the random number is larger, the StochSim concludes that no reaction occurs in this time step. Otherwise there is a bi-molecule reaction to fire. If there is only one possible reaction channel between

these two molecules, the reaction is selected to fire. Otherwise, the code selects one of the possible reaction channels in a similar way as (3.2) in the SSA.

6. Update the system accordingly if any reaction channel is selected. Proceed to the next time step.

By using the modified procedure above, the advantages are at least in two respects. Firstly, we do not have to introduce pseudo-molecules into the system, thus simplifying algorithm implementation. Secondly, since the simulation procedure has been different and we do not have $N_{psm:obj}$ any more, the possibility formula (3.4) and (3.5) have been replaced by (4.1) and (4.2) as follows. For a uni-molecule reaction,

$$p'_{1,j} = k_{1,j} N_{m:obj} \Delta t. \tag{4.1}$$

For a bi-molecule reaction,

$$p'_{2,j} = \frac{k_{2,j} N_{m:obj} (N_{m:obj} - 1) \Delta t}{2 N_A V}. \tag{4.2}$$

Consequently, (3.6) has also been changed as follows.

$$\Delta t = min(\frac{MAXP}{k_{1,max} N_{m:obj}}, \frac{MAXP}{\frac{k_{2,max} N_{m:obj} (N_{m:obj} - 1)}{2 N_A V}}). \tag{4.3}$$

As we can see that, the optimized Δt calculated from (4.3) is larger than the one derived from (3.6), thus making the algorithm more efficient.

However, one may argue about the inaccuracy caused by these modified formula. In step 3 and (4.1), the method actually assumes that, if any uni-molecule reaction were to fire, then no bi-molecule reactions would be considered to fire. Then this assumption would introduce error. Therefore, the possibility calculated from (4.2) is actually not precise. This is true, and the accurate form for bi-molecule reaction j should be:

$$p''_{2,j} = p'_{2,j}(1 - p'_{1,j}) = p'_{2,j} + O((\Delta t)^2), \tag{4.4}$$

where $p'_{1,j}$ is the possibility calculated in step 3, and $p'_{2,j}$ is the possibility calculated in step 5. As we can see, the difference between (4.2) and (4.4) is $O((\Delta t)^2)$. According to the calculation shown in (3.9), the error caused by (4.4) is in the same order as the error of the original StochSim is, therefore it will not cause loss of accuracy.

4.1.2 Combination of SSA and StochSim

With the improved generation strategy of the time step size, we are able to combine the StochSim with the SSA. Our main idea is to partition the complicated system into two subsystems, one for SSA and the other for StochSim. The whole reaction set is partitioned into two subsystems, according to the types of the reactants in each reaction. Suppose the system has N_n non-multi-state species $\{Sn_1, \ldots, Sn_{N_n}\}$, and N_m multi-state species $\{Sm_1, \ldots, Sm_{N_m}\}$. The system can involve three types of reaction channels:

1. *nn reactions*: uni-molecule reactions for one non-multi-state species and bi-molecule reactions between two non-multi-state species;

2. *mn reactions*: uni-molecule reaction for one multi-state species and bi-molecule reactions between one non-multi-state-species and one multi-state species;

3. *mm reactions*: bi-molecule reactions between two multi-state species.

Before simulation, the system is divided into the two following subsystems in terms of the descriptions below:

1. The SSA subsystem: this subsystem contains all nn reactions, with only non-multi-state species as reactants. This subsystem can be simulated by the SSA, which keeps a state vector of populations for all non-multi-state species.

2. StochSim subsystem: this subsystem contains all reactions involved with at least one multi-state species, namely all mn reactions and mm reactions. This subsystem can be simulated by our improved version of the StochSim, in which the state vector represents the molecules.

After the division, the hybrid scheme follows the principle of the first reaction method version of the SSA, to find *the next occurring reaction*. Basically, in each time step, each subsystem generates a time at which its next reaction will occur. The two times are compared to figure out which reaction in which subsystem should fire first. However, we must note that, the

StochSim subsystem does not actually predict the time in which the next reaction will occur, but for a time interval it decides whether or not there is a reaction firing.

Suppose current time is t. If the SSA subsystem time denoted by τ is smaller than the StochSim subsystem time denoted by Δt, it is important to figure out in time interval $[t, t+\tau)$ whether there is a reaction to fire in StochSim subsystem, because no reaction will fire in the SSA subsystem in this interval. Then we proceed the StochSim subsystem with a step size of τ instead of Δt, knowing that reducing Δt does not harm the accuracy of the StochSim. If there is a reaction to fire in the StochSim subsystem, then it is the next occurring reaction. Otherwise the reaction selected in the SSA subsystem at time $t+\tau$ is the next occurring reaction.

If τ is larger than or equal to Δt, we should figure out in time interval $[t, t+\Delta t)$ whether there is a reaction firing in the StochSim subsystem. Then we proceed the StochSim subsystem with a step size of Δt. If there is a reaction to fire, then it is the next occurring reaction. Otherwise, let $\tau = \tau - \Delta t$, and redo the comparison between τ and Δt to continue looking for the next occurring reaction.

Therefore, we have the simulation proceed as follows:

1. The SSA subsystem generates τ.

2. The StochSim subsystem generates an adaptive step size Δt.

3. Compare τ with Δt.

4. If τ is smaller, proceed StochSim subsystem with a step size of τ and let $t = t + \tau$. If any reaction occurs, go back to step 1. Otherwise, generate j, fire Reaction j in the SSA subsystem, and go back to step 1.

5. If τ is larger, proceed the StochSim subsystem with a step size of Δt and let $t = t + \Delta t$. If any reaction occurs, go to step 1. Otherwise, let $\tau = \tau - \Delta t$ and go to step 3.

The procedure is also illustrated in Figure 4.1.

By dividing the system into two parts, the hybrid method actually prevents the SSA subsystem from dealing with multi-state species, meanwhile preventing the StochSim subsystem from dealing with non-multi-state species with usually high populations. In this way, both subsystems take advantages of their own and avoid their drawbacks.

4.2 Numerical Experiments

To test the accuracy and efficiency of the hybrid method, we have constructed a toy model with several non-multi-state species and one multi-state species. Simulations are done with a Linux workstation based on Intel Pentium 4 CPU of 2.80GHz. The system is composed of the following reactions:

$$A + C \rightleftharpoons D \rightarrow P,$$

$$B + C \rightleftharpoons E \rightarrow Q,$$

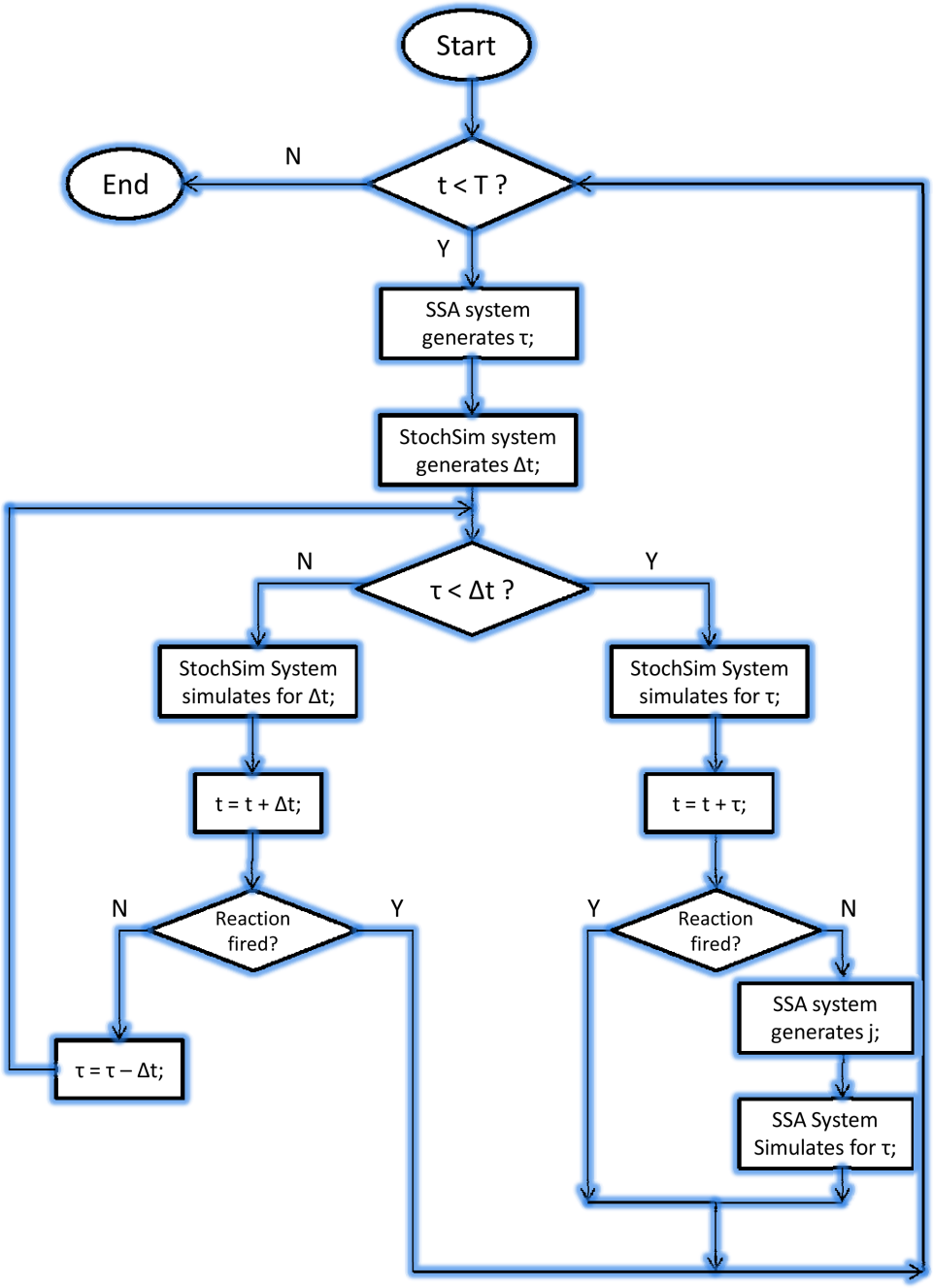


Figure 4.1: Flowchart of simulation procedure of each time step in the hybrid method

$$A \overset{F_n}{\rightleftharpoons} B,$$

$$F_n \rightleftharpoons F_{n-1},$$

$$F_n \underset{Q}{\overset{P}{\rightleftharpoons}} F_{n-1},$$

where F is the multi-state species and F_n represents F in state n. Consider F as an enzyme with a number of different states and each state has a different reaction rate constant for its enzyme ability. In this model, the rate constant is linearly related to the state number. To simulate this system with the hybrid scheme, all reactions involved with F are considered mn reactions, thus included in the StochSim subsystem, and all other reactions are nn reactions, included in the SSA subsystem. This model does not contain any mm reaction yet.

We analyze the efficiency of the hybrid method, the SSA and the StochSim from three perspectives to see how different features of the system affect the performance. In three groups of experiments, we respectively vary the populations of all the non-multi-state species, the number of the states of F and the population of F. The results are shown in Table 4.1-4.3. We can see that the hybrid scheme generally succeeds in combining the advantages of the SSA and the StochSim and in overrunning both of them in all situations.

Then we modify the model by replacing the reverse reaction in the last row

$$F_n \xrightarrow{Q} F_{n+1},$$

with the two following reactions

Initial Populations of S_{nms}	1	10	100	1,000	10,000
Hybrid	0.266	0.358	1.164	10.125	100.93
SSA	2.732	3.481	5.208	121.96	1217.9
StochSim	0.249	0.508	7.377	370.61	5082.4

Table 4.1: Numerical result comparison 1 among the hybrid method, SSA and StochSim. S_{nms} indicates all non-multi-state species including A, B, C, D, E, P and Q. The numbers in the table are simulation seconds for 1,000 runs on the model above by different algorithms.

Number of States of F	3	10	100	1,000	10,000
Hybrid	1.127	1.160	1.264	1.155	1.163
SSA	1.329	2.045	13.127	123.76	1339.2
StochSim	6.769	7.035	6.845	7.010	7.718

Table 4.2: Numerical result comparison 2 among the hybrid method, SSA and StochSim. The numbers in the table are simulation seconds for 1,000 runs on the model above by different algorithms.

$$F_n \xrightarrow{G} F_{n+1},$$

$$G_n \xrightarrow{Q} G_{n+1},$$

where G is also a multi-state species with a state of n. Note that the second reaction is an mm reaction. Similar experiments were conducted and we obtained similar results as above.

Numerical experiment results demonstrate that the new method has much better performance than either of the original algorithms for systems with both multi-state species and

Population of F	1	10	100	$1,000$	$10,000$
Hybrid	0.580	0.578	1.125	5.813	52.163
SSA	5.857	6.362	13.062	68.589	596.53
StochSim	5.300	5.228	6.692	22.730	637.98

Table 4.3: Numerical result comparison 3 among the hybrid method, SSA and StochSim. The numbers in the table are simulation seconds for 1,000 runs on the model above by different algorithms.

a large total population.

Chapter 5

Rule-Based Modeling and Simulation

Besides the StochSim, another way to deal with the multi-state situation is to use rule-based modeling [34, 35]. In this chapter, we first introduce the idea of this modeling method and one simulation method called the network-free algorithm. Then, we propose two new methods for simulating rule-based models. Last, numerical results are presented.

5.1 Background

5.1.1 Rule-based modeling

Rule-based modeling uses rules to replace reaction channels. A rule defines how a molecular particle reacts with other particles, depending on the states of their binding sites. Suppose protein species A has three binding sites and the first site can either be open or occupied

by a protein of species B. A rule might include all reactions that bind a B protein onto the first site of an A protein, regardless of the other two sites:

$$A_{open,?,?} + B \xrightarrow{k_{rule}} A_{B,?,?} \cdot B, \tag{5.1}$$

where the subscripts of a species describe the configuration of all its binding sites. The dot connecting A and B on the right-hand side of the rule indicates a bond. A question mark indicates all possible states for a certain site. k_{rule} is the rule rate constant. To represent this single rule, the SSA must assign one reaction channel to every qualified state variable of species A and therefore may have a large system size.

5.1.2 Network-free algorithm (NFA)

Due to combinatorial complexity, when converted into the SSA structure a rule-based model usually has a large system size. The particle-based NFA [18] was proposed for rule-based models. The NFA calculates the propensities for rules in a similar way as in the SSA. The major difference is that, instead of using populations of the reactant species, it uses populations of all particles that are candidates to be the reactants of a rule. For example, bi-molecule rule (5.1)'s propensity is calculated by the reaction rate k_{rule} times the number of particles of species A with the first site open, then times the number of particles of species B. The firing time and index of the next reaction are still calculated as in (3.1) and (3.2). For each rule, reactant candidate lists are created to store references to all particles that satisfy

the rule conditions. Thus when a rule is selected as in the SSA procedure, reacting particles in the selected rule are selected from corresponding candidate lists by generating uniform random numbers to calculate the indices of the reacting particles. The general procedure of the NFA for our implementation is as follows.

Initialization In general, every rule entails the following information.

1. The description of the reactants and products with their species and states of all the binding sites, including the binding site conditions for this rule to fire.
2. Rule rate constant, by default each rule has its own rate.
3. Lists of candidate reactants. To facilitate selecting reacting particles after the rule index is determined, the method maintains a reactant list for each reactant in each rule. A reactant list contains references to all the candidate particles that meet the corresponding binding site conditions in its rule.
4. Populations of reactants. To calculate propensities quickly, populations of reactants are recorded and updated when necessary, so that the counting of particles in reactant lists is not repeated in every step.

The algorithm first creates rules by initializing items (1) and (2). Then objects for all molecular particles in the system are created. Upon the creation of each object, all rules are scanned. If the created object qualifies as a reactant of some rule, a reference to the

object is added to the corresponding reactant list and the reactant population is increased by one. After all particle objects are created and assigned, initialization of items (3) and (4) is completed.

Simulation The NFA repeats the following simulation procedure until a stopping criterion is reached.

1. Calculate propensities for all rules, using the rules' reactant populations and rate constants.

2. Calculate the next event time with equation (3.1).

3. Select a rule index with equation (3.2).

4. Select reacting objects (particles) from the corresponding reactant lists of the selected rule. In this step, the index of each reacting particle in the corresponding list is calculated by generating a uniform random number and multiplying it with the total population in the list.

5. Update the system by executing the selected rule. Destroy objects of reactants and create products, if necessary, as described by the rule. For a reactant object, remove all of its references from corresponding reactant lists and decrease the corresponding reactant population by one. Then search all rules for the created object. If it qualifies for a rule, its reference is inserted into the corresponding reactant list and the reactant population is increased by one.

6. Update time, and go to Step 1.

The major advantage of the NFA is that it does not generate small time steps as the StochSim often does, and it does not need to generate the huge reaction network that Gillespie's SSA does. The disadvantage is the extra cost for the storage of a large number of particle objects, and the maintenance of the reactant lists of all rules in every time step.

5.2 New rule-based simulation methods

Since the NFA creates objects for all molecular particles and keeps reactant lists for each rule, if the total population in the system is large, NFA incurs a severe memory and time cost for the maintenance of those long reactant lists. With these concerns in mind, we modify NFA to reduce its space and time complexity as much as possible.

5.2.1 Population-based NFA (PNFA)

Here we propose a hybrid population-based NFA, which stores single-state species as populations instead of creating multiple individual objects. In other words, the PNFA creates only one object for each single-state species with a population. Objects of this type are called *population objects*. The difference between population objects and normal molecular objects is that population objects store the populations of the species while molecular objects store the state of binding sites. With this change, the space complexity of NFA is greatly re-

duced for those systems involving simple chemical species with high populations. Of course, changes to the state variables require corresponding modifications in the simulation steps. We highlight the changes from the NFA to the PNFA procedure.

Initialization The algorithm first creates rules by initializing their definitions of reactants, products, and rate constants. Then objects for all molecular particles in the system are created, species by species. For a single-state species, a population object is created with its initial population. For a multi-state species, normal molecular objects are created with binding site states. Upon the creation of each object, all rules are scanned. If the created object qualifies as the reactant of some rule, a reference to the object is added to the corresponding reactant list and the reactant population is increased by one if the created object is a normal molecular object (multi-state species). Population objects increase the reactant population by the amount of the object's population. After all objects are created, initialization is completed.

Simulation The simulation procedure is similar to that for the NFA, except that the PNFA treats population objects and normal molecular objects differently. We replace the original Step 4 with the following.

4*: Select reacting objects from the corresponding reactant lists of the selected rule. In most cases, the reactant list contains either a population object or molecular objects, but not both, in which case selection is easy. When both exist in the list, the number of molecular

objects is calculated and treated as a pseudo population object. We first search for which (population) object is selected. If the selection is a normal population object, this object is chosen. Otherwise we use a uniform random number to select among the molecular objects of the chosen pseudo population object.

We replace Step 5 with the following.

5*: Update the system by executing the selected rule. Destroy objects of reactants and create products as described by the rule. For a molecular object, the operation is the same as in the original Step 5. A population object needs only to decrease or increase its population value and reactant populations for related rules.

5.2.2 Full-scale SSA (FSSSA)

The PNFA is a hybrid scheme combining both population-based and particle-based strategies. It is natural to extend the idea and ask: "Is the particle-based scheme always efficient for simulating rule-based models? Under what condition will a pure population-based method perform better for a rule-based model?" We believe that, besides single-state species, any multi-state species with a large population and a small number of states would be better represented by a population instead of as many particles. On the other hand, the particle-based scheme is preferred when a small population of a multi-state species is sparsely distributed into a large number of states.

Moreover, for conventional population-based simulation schemes applied on multi-state mod-

els, the number of reaction channels usually scales with the number of present variables. Therefore, the drawback of a pure population-based scheme is the possibly huge system size. Note that in a real simulation, a multi-state species may not access all its possible states. This feature results in many state variables with zero population in the system. Sparse storage on population objects is thus a natural solution. Only objects with non-zero populations need to be present in the system in the simulation. Similar technique has been used in Moleculizer developed by Lok et al. [58]. The difference is that Moleculizer generates the reaction network dynamically while here we will adopt the network free design and stick with rules.

With the features mentioned above, we propose the full-scale SSA (FSSSA), a totally population-based algorithm for simulating rule-based models. The FSSSA can also be considered as a new implementation of the SSA on rule-based models. Here is the FSSSA simulation procedure.

Initialization Like the NFA, rules are first initialized with their first two data items. Species are initialized one by one. A single-state species is simply treated as a multistate species with one state. For each state of each multistate species, if it has a non-zero initial population, an object is created to store its population and binding site states. Then reactant lists and populations of rules are created in the same way as in the PNFA.

Simulation The simulation procedure is similar to that for the NFA, but without molecular objects. As a result, we replace the original NFA Steps 4 and 5 with the following two steps.

4**. For a selected rule, select a particular object for each reactant. The search is done within the reactant lists of the rule, which store references to objects with non-zero populations. The selection procedure is similar to SSA equation (3.2).

5**. Update the system by executing the selected rule. For each selected reactant object, decrease its population by one. If its population decreases to zero, remove all of its existing references from any reactant lists and destroy this object. Decrease the reactant populations of related rules by one. With the description of each product (species and binding site states), search for the product object. If the product object already exists, simply increase its population by one. If the product object does not exist, create an object with population of one and its binding site states, scan all the rules, and insert its references onto qualified reactant lists. Increase the reactant populations of related rules by one.

5.2.3 Complexity analysis

Except for the StochSim, the algorithms described in this chapter can all be considered as different implementation strategies for the SSA. The key difference is in their efficiency. Below we offer a complexity comparison of these algorithms.

Similarly to the notations in previous chapters, we suppose N is the number of species, M is the number of reaction channels, N_R is the number of rules in the system, N_S is the maximum

number of states for each multi-state species S, $N_{m:obj}$ is the total number of molecular objects in the system, and $N_{pm:obj}$ is the total number of pseudo molecular (molecular and population) objects in the system. Table 5.1 gives the time and space complexities of four simulation methods for rule-based models. Note that since these four methods all use (3.1) to generate the next reaction time, their average numbers of time steps are asymptotically the same. Therefore, the computational cost of each time step can be divided into three parts: reactant selection, system update, and propensity calculation.

	Time Complexity of Each Step			Space
	reactant selection	system update	propensity calculation	Complexity
SSA	$O(\log(M))$	$O(1)$	$O(\log(M))$	$O(N)$
NFA	$O(N_R)$	$O(N_R)$	$O(N_R)$	$O(N_{m:obj})$
PNFA	$O(N_R)$	$O(N_R)$	$O(N_R)$	$O(N_{pm:obj})$
FSSSA	$O(N_R) + O(\hat{N})$	$O(N_R)$	$O(N_R)$	$O(N\hat{N})$

Table 5.1: Complexity of four methods for rule-based models. $\hat{N} = \max_{s \in S} N_s$.

Note that for the original SSA, the time complexity can be achieved in time $O(\log(M))$ with carefully designed data structures such as reaction dependent graph and reaction time priority queue [26]. For a large system, $\log(M)$ is comparable with N_R. But the overhead resulted from those data structure is quite large [27]. Usually the original SSA still exhibits lower efficiency than the other three methods. Comparing the NFA and the PNFA, since the number of rules is not large, linear search among rules is actually more efficient. Thus

for the NFA and the PNFA their time complexities for selecting rule to fire are $O(N_R)$. For the system update part, there is a good chance for the PNFA to select population objects. Since the PNFA updates a population object without removing or inserting references on corresponding reactant lists, it often has a smaller runtime constant than the NFA. Therefore, its total runtime is often lower than that for the NFA. $N_{pm:obj}$ for the PNFA is different from $N_{m:obj}$ for the NFA. The space complexity of the PNFA is lower than that of the NFA because of the population objects, which can make $N_{pm:obj} \ll N_{m:obj}$.

Comparing the PNFA and the FSSSA, since the latter still uses (3.2) to select the states of the reactants from reactant lists, its reactant selection part has an extra term $O(\hat{N})$. For the system update part, again, since updating population objects has less operations than updating molecular objects, the runtime constant of the FSSSA can be smaller than that of the PNFA. Therefore, when $\hat{N}$ is not large compared with N_R, the FSSSA may need less total time. On the other hand, if $\hat{N}$ is large, FSSSA may not be efficient. For a system with a few molecules that could have great numbers of states, and coupled with uni-state species with large populations, the hybrid simulation scheme PNFA will show its advantage over both FSSSA and NFA. However, we should note that this is also the case where most of the states of the multi-state species have zero populations. Then we should still expect savings due of our sparse storage implementation on population objects.

Between the SSA and the FSSSA, the comparison of the asymptotic space complexity depends on $\hat{N}$. If $\hat{N} = O(1)$, the FSSSA generally consumes less actual memory than the SSA because of the smaller size of the rule-based model. The SSA uses elementary single-state

rules, resulting in a larger rule network.

5.3 Numerical experiments

To compare with the SSA the methods for rule-based models in both accuracy and efficiency, we present numerical experiments on the bistable switch model. In the following the experiments were performed on a 2.66 GHz Intel Core 2 Duo iMac with codes written by the authors in C++ languages. As shown in Figure 5.1, the model is a small but realistic example of a motif in regulatory models which shows an antagonistic relationship between Clb2 and Cdh1, mediated by the amount of catalyst Cdc14 [59]. Cdh1 activates the degradation of Clb2, where the catalytic rate associated with the unphosphorylated form of Cdh1 is much larger than those phosphorylated forms of Cdh1. Therefore, these two species are in opposition to one another. With careful selection of its parameters, the model can exhibit bistable behaviors. This model is actually the same bistable switch model based on multiple site phosphorylation as shown in Figure 1.4.

It can be modeled by the rule-based modeling method with the following rules:

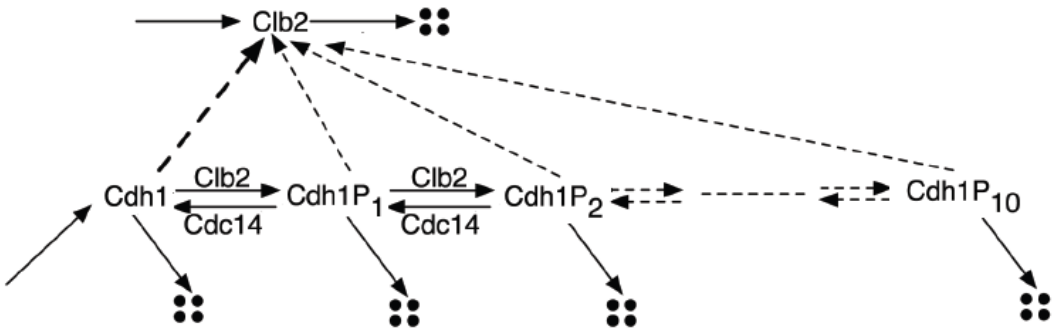


Figure 5.1: Bistable switch motif. Clb2-Cdh1 antagonistic interaction forms a bistable switch with Cdc14 as the control variable.

$$\phi \xrightarrow{k_{s1}} Clb2, \tag{5.2}$$

$$Clb2 + Cdh1P\{i\} \xrightarrow{k_{d1}} Cdh1P\{i\}, \tag{5.3}$$

$$Cdh1P\{i\} + Clb2 \xrightarrow{k_p} Clb2 + Cdh1P\{i+1\}, \qquad 0 \le i \le MAX - 1, \tag{5.4}$$

$$Cdh1P\{i\} + Cdc14 \xrightarrow{k_{dp}} Cdc14 + Cdh1P\{i-1\}, \qquad 1 \le i \le MAX, \tag{5.5}$$

$$\phi \xrightarrow{k_{s2}} Cdh1P\{0\}, \tag{5.6}$$

$$Cdh1P\{i\} \xrightarrow{k_{d2}} \phi, \qquad 0 \le i \le MAX. \tag{5.7}$$

All equations here are assumed to operate using the simple mass action rate law. The integer subscript i represents the level of phosphorylation ranging from 0 to MAX, where $5 \le MAX \le 20$ is a range for typical phosphorylation levels in budding yeast cells. We choose $MAX = 10$ for our experiments. The parameters are $k_{s1} = 0.4$, $k_p = 0.05$, $k_{dp} = 0.05$, $k_{s2} = 0.05$, $k_{d2} = 0.0025$. For reaction (5.3), the reaction rate is calculated by $k_{d1} =$

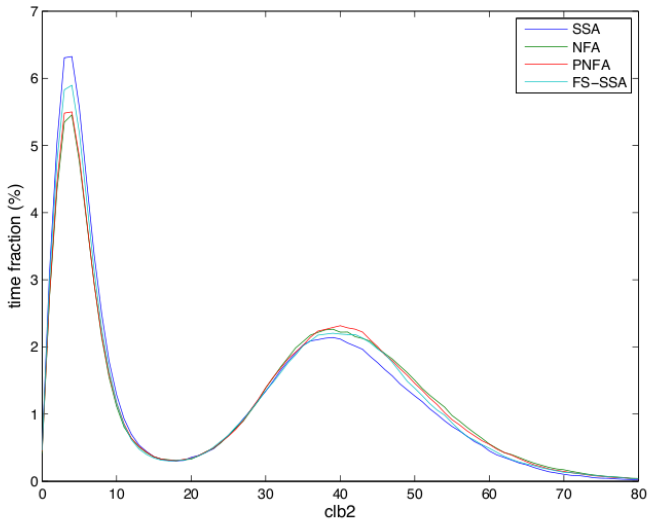


Figure 5.2: Bistable distribution comparison of bistable switch motif. Distributions of $Clb2$ by SSA, NFA, PNFA and FSSSA with $Cdc14 = 19.0$. Results are obtained with a simulation of 5×10^6 time units.

$0.0055 \cdot Cdh1P\{0\} + 0.0005 \cdot \sum_{i=1}^{MAX} Cdh1P\{i\}$. For initial values, we start with $Clb2$ and $Cdh1P\{i\}, i = 1, \ldots, 10$ all at 0 and $Cdh1P\{0\}$ at 10. As the parameter $Cdc14$ varies, the distribution of $Clb2$ exhibits mono-stable or bistable behavior.

Figures 5.2-5.4 compare the distributions and time trajectories of Clb2 generated by the SSA and the rule-based simulation methods. Since all the methods yield nearly identical distributions and similar system behaviors, we expect that they have comparable accuracy.

Table 5.2 shows runtimes for this model. To configure this model for the SSA, the six rules in 5.2 are translated to 34 reaction channels with 12 state variables involved. The SSA has a constant time complexity in system update, but it spends much more CPU time than the other three methods in calculating the propensities. The NFA has the worst performance due

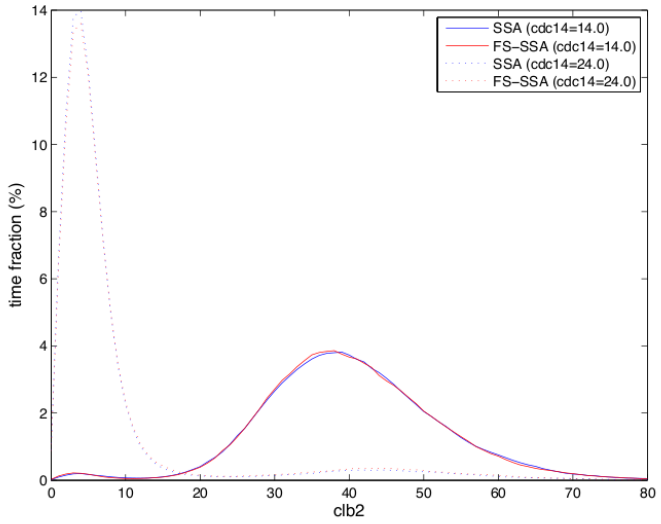


Figure 5.3: Mono-stable distribution comparison of bistable switch motif. Distributions of $Clb2$ by SSA, NFA, PNFA and FSSSA with $Cdc14 = 14.0$ and $Cdc14 = 24.0$. Results are obtained with a simulation of 5×10^6 time units.

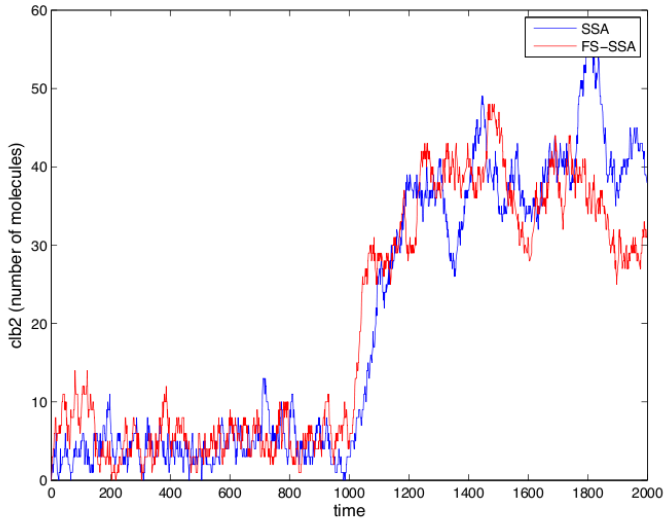


Figure 5.4: Time trajectory comparison of bistable switch motif. The trajectories show the transition from the lower steady state to the upper steady state due to a sudden drop in the level of Cdc14.

to its heavy overhead in system update for maintaining the data structure of the reactant lists. The PNFA is able to save a good amount of CPU time in this part thanks to the population-object for the single-state species Clb2. The FSSSA performs the best, because it saves even more by using population-objects for the multi-state species Cdh1. Therefore, we see for this model that it is more efficient to treat the multi-state species as populations than as particles.

	CPU time in seconds				
Method	reactant selection	system update	propensity calculation	other	total
SSA	31.5	4.0	50.3	15.2	101
NFA	30.9	281.2	11.2	19.7	343
PNFA	28.0	211.5	11.1	18.4	269
FSSSA	28.2	63.0	8.8	15.0	115

Table 5.2: CPU time comparison on bistable switch motif. CPU times used by four simulation methods on a Mac Core 2 Duo 2.0 GHz machine for simulation on the bistable switch model. Results were obtained with a simulation of 5×10^6 time units.

Part II

Multi-Scale Simulation Methods

Chapter 6

Hybrid Modeling and Simulation of Stochastic Effects on Progression through the Eukaryotic Cell Cycle

Recent discoveries have shown that the cell cycle is characterized by different types of randomness, which play important roles in cell physiology [2, 60]. Stochastic models of cell cycle regulation are necessary to understand these experimental results in quantitative terms. For our research on the stochastic simulation, they are also very interesting and realistic models with the multi-scale feature.

In this chapter, we propose to use Haseltine and Rawlings' hybrid method [22] to take advantage of the scale difference in Kar et al.'s cell cycle model [11], and accelerate the stochastic

simulation with a more efficient partitioning strategy. Numerical experiments demonstrated that, with our new partitioning strategy, the hybrid method accurately simulates intrinsic noise with significantly improved simulation efficiency.

We also build a stochastic cell cycle model based on Tyson and Novak's model [38]. By assuming most of the intrinsic noise come from the gene expression level, we build stochastic reactions at the gene expression level and preserve the original phenomenological ODEs from the deterministic model. The combined new model is self-partitioned and easily simulated by the hybrid method. Numerical results show that this model can produce cell cycle kinetics and variability which are consistent with experimental observations.

6.1 Motivation

The eukaryotic cell cycle is regulated by a complicated chemical reaction network. To model the cell cycle control system, theoretical biologists started with deterministic models using ordinary differential equations (ODEs) [61, 62, 63, 64, 65, 66, 38, 56, 4, 67]. Although deterministic cell cycle models can be precise and robust in many respects, experimental data exhibit considerable variability from cell to cell during cell growth and division [1, 54, 2]. For example, the coefficient of variation ($\mathrm{CV} = \frac{\text{standard deviation}}{\text{mean}}$) of size at division for fission yeast cells is around 7.5%, and the CV of their cell cycle time is up to 14% [1]. This observed noise is usually attributed to two sources: intrinsic noise, from fluctuations of molecule numbers present within a single cell; and extrinsic noise, from inequalities in sizes of the two daughter

cells after division. Given the small volume of a cell (e.g., a yeast cell is roughly 30 femtoliters at birth), the total number of molecules of a particular protein species is usually limited to several thousand. Moreover, the number of molecules of the mRNA for each protein at any time may be less than 10 [68]. In this case, molecular fluctuations cannot be neglected, and they may significantly affect the behavior of the cell. Therefore, to accurately model the cell cycle, stochastic models and simulations are required to capture this noise.

A natural way to build a stochastic model is to convert a deterministic model into its stochastic counterpart, which employs only elementary reaction steps, suitable for simulation by Gillespie's stochastic simulation algorithm (SSA) [13, 14]. One of the major difficulties in this type of conversion lies in the rate laws, which are often not elementary (mass-action) kinetics. For example, Tyson and Novak's simple three-variable model of the cell cycle [38] (see Section 6.2) involves variables X, Y, and Z. The phosphorylation and dephosphorylation of Y are governed by Michaelis-Menten rate laws, and the synthesis of Z is given by a Hill function. These phenomenological rate laws are derived from more detailed elementary reactions mechanisms using pseudo-steady state approximations. However, applying Gillespie's SSA to phenomenological rate laws may possibly generate incorrect stochastic results [55]. Thus a model based fully on mass-action kinetics for all reactions was developed in Kar et al. [11] by "unpacking" the reactions with phenomenological rate laws in the Tyson-Novak three-variable model into sets of elementary stochastic reactions. In the process, mRNA variables for X, Y, Z, and other helper proteins were introduced into the network. The unpacked model was then simulated using Gillespie's SSA. Kar et al.'s work managed to model the

repetitive cell cycle behavior on average and capture proper amounts of both the extrinsic and intrinsic noise. However, the cost is a much larger system of variables and reactions. If a more detailed deterministic model, such as Chen et al. [4], were to be unpacked for stochastic simulation, the complexity of the model would quickly increase as well as the CPU time for simulation by Gillespie's SSA.

Our goal is to develop a modeling and simulation strategy for cell cycle models that is both accurate and efficient. First, to improve the simulation efficiency of Kar et al.'s stochastic model, we apply Haseltine and Rawling's hybrid method [22] with different partitioning strategies, and check the corresponding accuracy and efficiency. We demonstrate that, with a more efficient partitioning strategy, the hybrid simulation gives reasonably accurate results and saves significant computational cost as compared to the full stochastic simulation. Furthermore, through an analysis of the partitioning results in the hybrid simulation, we conclude that a good partitioning strategy for a cell cycle model is to treat all reactions related to gene expression (reactions modifying gene and mRNA species) as "slow" reactions. Biologically, it matches with our intuition that (for cell cycle models) most of the intrinsic noise arises at the gene expression level due to the low numbers of molecules of genes and mRNAs. On the other hand, it also implies that, to prepare a deterministic cell cycle model for stochastic simulations, one may not need to unpack the phenomenological rate laws. Instead, we managed to build a new hybrid model by packing Kar et al.'s model back to a comparable three-variable model. The ODEs are similar to the original three-variable model, while stochastic elementary reactions related to gene expression are added as slow reactions

and simulated with the SSA. In this way, we avoid creating a large reaction network while obtaining sufficient accuracy and improved efficiency in simulation.

6.2 Cell Cycle Models

The cell cycle is driven by the mutual antagonism between B-type cyclins (such as Clb2) and G1-stabilizers (such as Cdh1) [69]. When B-type cyclins are abundant, they combine with kinase subunits (Cdk1) to form active protein kinases (e.g., Cdk1-Clb5 and Cdk1-Clb2 in budding yeast) that promote DNA synthesis and mitosis (S, G2, and M phases of the cell cycle). When Cdh1 is active, Clb-levels are low, and cells are in the unreplicated phase (G1) of the DNA replication-division cycle. The cell cycle control system alternates back-and-forth between G1 phase (Cdh1 active, Clb-levels low) and the S-G2-M phase (Cdh1 inactive, Clb-levels high).

Tyson and Novak built their three-variable model [38] based on a bistable switch created by the antagonism between Cdk1-Clb complexes, denoted by X, and Cdh1, denoted by Y, as illustrated in Figure 6.1.

In this model, X inactivates Y by phosphorylating it, and the unphosphorylated Y catalyzes the degradation of X. The exit protein, Cdc20, activates a phosphatase (Cdc14) that dephosphorylates and activates Y. The actions of Cdc20 and Cdc14 are lumped together in a single variable Z, whose synthesis is promoted by X. Therefore, X is involved in a positive feedback loop with its repressor Y (mutual antagonism), which creates a bistable switch.

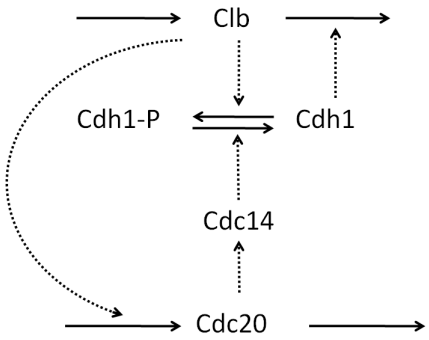


Figure 6.1: Bistable switch on which Tyson and Novak's model is based

Cell growth flips the switch from the G1 state to the S-G2-M state. The reverse transition (back to the G1 state) is triggered by the negative feedback loop in the model (X activates Z, Z activates Y, and Y inactivates X).

This three-variable model can be formulated as the following set of ODEs [38]:

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{dt}}V &= \mu V, \\ \frac{\mathrm{d}}{\mathrm{dt}}[\mathrm{X}] &= k_{\mathrm{sx}}V - (k_{\mathrm{dx}} + k_{\mathrm{dxy}}[\mathrm{Y}])[\mathrm{X}], \\ \frac{\mathrm{d}}{\mathrm{dt}}[\mathrm{Y}] &= \frac{(k_{\mathrm{hy}}+k_{\mathrm{hyz}}[\mathrm{Z}])([\mathrm{Y_T}]-[\mathrm{Y}])}{J_{\mathrm{hy}}+[\mathrm{Y_T}]-[\mathrm{Y}]} - \frac{k_{\mathrm{pyx}}[\mathrm{X}][\mathrm{Y}]}{J_{\mathrm{pyx}}+[\mathrm{Y}]}, \\ \frac{\mathrm{d}}{\mathrm{dt}}[\mathrm{Z}] &= k_{\mathrm{sz}} + \frac{k_{\mathrm{smzx}}[\mathrm{X}]^n}{{J_{\mathrm{smzx}}}^n+[\mathrm{X}]^n} - k_{\mathrm{dz}}[\mathrm{Z}], \end{cases} \tag{6.1}$$

where square brackets denote the concentration of a chemical species. The k's are reaction rate constants, and the J's are equilibrium binding constants (units = concentration). $[\mathrm{Y_T}]$ is the total concentration of Y (the sum of the phosphorylated and unphosphorylated forms). The Hill exponent, n, determines the steepness of the sigmoidal curve expressing the dependence of Z-synthesis on [X]. With proper choice of parameter values, this model

shows alternations of G1 phase and the S-G2-M phase in deterministic simulations. A more detailed deterministic cell cycle model for budding yeast, developed by Chen et al. [4], successfully accounts for the phenotypes of wild-type budding yeast cells and $\sim$ 120 mutant strains.

The equations (6.1) represent a deterministic phenomenological model. In order to build a stochastic cell cycle model, the state variables have to be converted from concentrations (nM) to numbers of molecules per cell ("populations"), and new state variables have to be added to represent mRNA populations, because intrinsic noise in cells is mostly generated at the gene expression level [70, 71, 72]. Moreover, following Gillespie's framework of SSA, chemical reactions with mass action kinetics are desired to capture the full stochastic effect in chemical reaction systems. Thus phenomenological rate laws, such as Michaelis-Menten and Hill functions, need to be replaced by elementary mass-action kinetics with additional variables as intermediate species, which results in a much larger system of equations. With all these considerations, Kar et al. [11] built a stochastic cell cycle model with 19 variables and 47 reactions, based on Tyson and Novak's three-variable model. Kar et al.'s model generates distributions of cell cycle times and division sizes that match the data observed in wet-lab experiments. However, conversion from the original three-variable, phenomenological, deterministic model to the fully unpacked and supplemented stochastic model takes a great amount of work, and simulations of the unpacked model by Gillespie's SSA are slow. The goal of this paper is to reduce the modeling effort and to improve the efficiency of the stochastic simulation without sacrificing accuracy of the model.

6.3 Hybrid Method and Partitioning Strategies

6.3.1 Hybrid Method

For large systems with fast reactions, Gillespie's SSA can be computationally slow because it simulates every reaction event. To speed up the SSA, several approximate simulation strategies have been proposed. One group of approximation methods tries to take advantage of the multiscale characteristics observed in the reactant populations. Some species are present at larger population numbers than others. If all reactants have relatively large populations, one can represent a stochastic system by chemical Langevin equations (CLEs) and solve them as stochastic differential equations [46], or directly apply tau-leap methods [47, 48], which approximate the numbers of reactions by Poisson random numbers. The other group of approximation methods tries to take advantage of multiscale features in the reactions: some reactions occur much more frequently than others (these are called "fast" reactions, in contrast to "slow" reactions that occur comparatively infrequently). For fast reactions, a quasi-steady state assumption on the fast variables [19] or a partial equilibrium assumption on the fast reactions [20] can be applied to reduce the system and accelerate the simulation. In realistically large biochemical systems, usually both multiscale features are present. Species populations and reaction propensities can span several orders of magnitude. Thus it is not realistic to simulate a multiscale system with only one method that works well in one scale. Instead, hybrid methods should be considered from a more practical, system-specific point of view.

Several hybrid methods have been presented [22, 53, 21]. Cao et al. [52] proposed to partition the system based simply on the species population numbers. For species whose population numbers are less than a threshold, all related reactions are simulated by SSA, while other reactions are simulated by the tau-leaping method. Haseltine and Rawlings [22] proposed to partition a system into groups of slow and fast reactions. The partitioning criterion is determined by two thresholds set by the user before simulation. A reaction is put into the fast reaction group if its propensity is greater than the propensity threshold and the populations of all its reactants are greater than the population threshold. In this method, the fast reaction group is governed by ODEs or CLEs and the slow reaction group is simulated by Gillespie's direct method. A similar strategy was adopted by Salis et al. [53, 73], but fast reactions are approximated by CLEs and slow reactions are simulated by Gibson and Bruck's next reaction method [26]. They also developed a more efficient mechanism to monitor the occurrences of slow, discrete events while simultaneously simulating the dynamics of a continuous, stochastic or deterministic process.

Our work follows the original idea of the hybrid method from Haseltine and Rawlings [22], and adopts a similar implementation strategy for event handling as in Salis et al. [53, 73]. Suppose the system has N species, and its state vector is denoted by $X(t) = (X_1(t), \ldots, X_N(t))$, where $X_i(t)$ is the number of molecules of the i-th species at time t. Suppose M reactions are involved. The M reactions are partitioned into two subsets: S_{fast} for fast reactions, which are formulated by ODEs, and S_{slow} for slow reactions, which are stochastic reactions. Let $a_i(x, t)$ be the propensity of the i-th reaction in S_{slow} when $X(t) = x$, let τ be

the jump interval of the next stochastic reaction, and let μ be its reaction index. Form the ODE system from the fast reaction set (S_{fast}) and the SSA system from the slow reaction set (S_{slow}). Set $t = 0$. The hybrid algorithm is given as follows:

Hybrid Simulation Algorithm

1. Generate two uniform random numbers r_1 and r_2 in $U(0, 1)$.

2. Integrate the ODE system with the integral equation:

$$\int_t^{t+\tau} a_{\text{tot}}(x, s)ds + log(r_1) = 0, \tag{6.2}$$

   where $a_{\text{tot}}(x, t)$ is the total propensity of S_{slow}.

3. Determine μ as the smallest integer satisfying

$$\sum_{i=1}^{\mu} a_i(x, t) > r_2 a_{\text{tot}}(x, t). \tag{6.3}$$

4. Update $X(t)$ according to the μ-th reaction in S_{slow}.

5. If stopping condition is not reached, go to Step 1.

Note that solving equation (6.2) is an important step, particularly when the slow reaction propensities change appreciably over time according to the fast reaction dynamics. The original strategy by Haseltine and Rawlings [22] is to add a propensity of "no reaction" that decreases the time of the next slow reaction, τ, so that the slow reaction propensities do not appreciably change over the time τ, while Salis et al. [53, 73] introduced a system of

jump differential equations, where each equation describes the propensity for a single slow reaction to occur. These jump equations are numerically integrated alongside the system of ODEs (or SDEs); when the solution to a jump equation crosses zero, its corresponding slow reaction has fired. This step is the key difference between the two algorithms. In our implementation, we follow a similar strategy as Salis et al. but apply it to the direct method instead of Gibson and Bruck's next reaction method [26]. Suppose that the ODE system is given by

$$x' = f(x). \tag{6.4}$$

We simply add an integration variable z and add an equation

$$z' = a_{\text{tot}}(x), \quad z(0) = 0. \tag{6.5}$$

In every simulation step, starting from time t, the ODEs (6.4) and (6.5) are numerically integrated until $z(t+\tau) = z(t) + log(r_1)$. Then τ gives the solution for the equation (6.2). Such an integration can be conveniently achieved through standard ODE solver with root-finding function such as LSODAR[74].

6.3.2 Partitioning Strategy

The hybrid simulation algorithm is straightforward if the system is well partitioned. However, an important decision has to be made on the partitioning strategy. The original partitioning strategy proposed by Haseltine and Rawlings, which was adopted in the software package hy3S [53, 73], partitions the system with two thresholds pre-selected by the user. One is

the population threshold $\bar{x}$, while the other is the propensity threshold $\bar{a}$. A reaction is considered as a fast reaction only when the populations of all its reactants are greater than $\bar{x}$ and its propensity is greater than $\bar{a}$. Figure 6.2 illustrates this strategy.

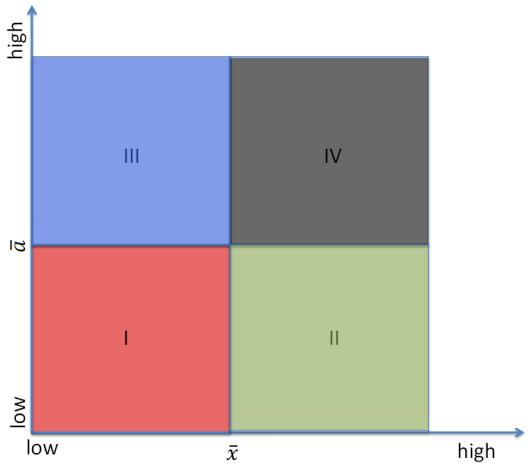


Figure 6.2: Scales of reactions and populations. Region I contains slow reactions whose reactants have low populations, region II contains slow reactions whose reactants have high populations, region III contains fast reactions whose reactants have low populations, while region IV contains fast reactions whose reactants have high populations.

Here the scales are measured by the time average of the populations and propensities. The species populations and reactions are sorted from low to high. The whole system is thus divided into four regions, as illustrated in the figure. Haseltine and Rawlings' partitioning strategy puts regions I, II, and III into the SSA regime and region IV into the ODE regime. This is a conservative strategy. Unfortunately, simulations for both the ODE and SSA regimes have to stop and restart for every SSA firing. Thus the efficiency of the hybrid method depends heavily on the stepsize allowed by the SSA, which is limited by the scales of

the reactions in the SSA regime. Particularly, the reactions in region III, which contains fast reactions with at least one reactant/product with a low population, will force the system to take small steps. Thus the efficiency of the hybrid method is limited by region III. This presents a challenge when the hybrid method is applied to model and simulate complex systems. In this paper we propose a different partitioning strategy for the cell cycle model: We only put region I into the SSA regime, while reactions in regions II, III, and IV are all simulated with ODEs.

6.3.3 Applying the hybrid method to the Cell Cycle Model

In preparing the Tyson-Novak three-variable model for stochastic simulation, Kar et al. not only converted the phenomenological rate laws (Michaelis-Menten and Hill kinetics) to elementary steps (mass-action kinetics) but also added stochastic reactions related to gene expression. We predict that most of the intrinsic noise in the cell cycle should come from the gene and mRNA species. These species have lower molecule numbers and react less frequently than protein level reactions. Meanwhile, the majority of the reactions are on the protein level, where insignificant noise is expected. These two levels suggest a natural partitioning of the system. But all these conjectures need verification from numerical results.

To find an appropriate partitioning of this system, we first generated a sample SSA run on the fully stochastic model to collect scale information for all the reactions from the simulation profile. Two characteristics are calculated and plotted in the sample run. One is the time

average population for each species. The other is the total number of firings for each reaction. (In the SSA, the total number of firings, which is easy to track, is equivalent to the scale of the integral of the propensities.) Figure 6.3 shows (on a double-log plot) the profile of all 47 reactions in Kar et al.'s model in terms of these two characteristics. Each circle represents a reaction. Some circles may overlap with others due to the resolution of the image. Note that for each reaction, its population scale is determined by the smallest time average population of all its reactants and products, but not catalysts.

We test four partitioning strategies as illustrated in Figure 6.3. Starting from the lower left part of the profiling plot, we initially partition the five slowest reactions with the lowest molecule numbers into the SSA system (strategy A). These reactions fire less than 10,000 times in 200 cycles (total number of reaction events $\approx 10^9$) and their reactants have average populations less than 1. Three of them control the activation/inactivation of the gene and two of them are synthesis and degradation of mRNA for protein Z. We predict that these reactions should be the main source of intrinsic noise. In strategy B, we add nine more reactions into the SSA system. These reactions fire less than 10^6 times in 200 cycles, and their reactants have average populations less than 10. They are all on the gene expression level. Eight of them are mRNA synthesis and degradation reactions and one is related to degradation of the transcription factor. Strategies A and B both follow the idea of partitioning only region I into the SSA regime. The difference is only on the threshold values. Interestingly, in strategy B the SSA regime includes all the reactions related to gene expression. For comparison purpose, we also tried two more partitioning strategies. Strategy

C includes ten more slow reactions ($< 10^6$) with high populations (> 30), while strategy D includes two more fast reactions ($> 10^6$) with low populations (< 10).

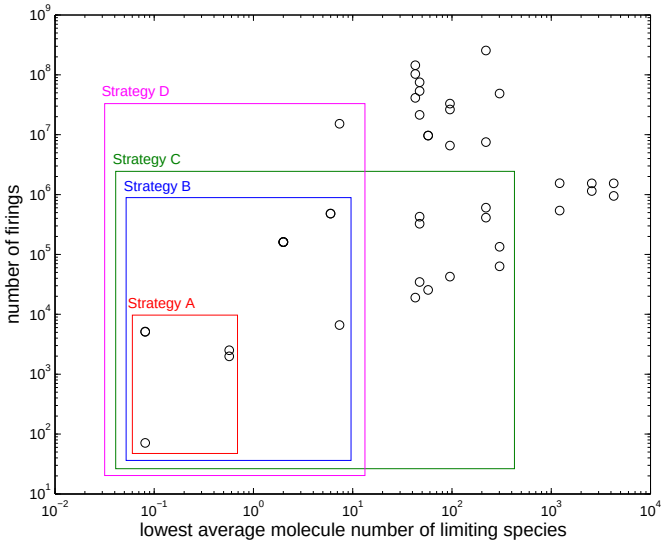


Figure 6.3: Profile of reactions in Kar's cell cycle model. Data are collected from a sample SSA run for continuous 200 cell cycles. Four partitioning strategies A, B, C and D are represented. For each of them, the reactions in the box are for the SSA system while the ones outside the box are for the ODE system.

Since the mRNAs for proteins X, Y and Z are crucial indicators of this cell cycle system, in order to analyze the impact of the four partitioning strategies, we first compare the hybrid method to the SSA on these distributions. Figure 6.4 and 6.5 show such comparison for strategies A and B, respectively. For strategy B (Figure 6.5), both methods generate nearly identical distributions of the mRNAs because all reactions related to the mRNAs are treated stochastically. For strategy A (Figure 6.4), only the mRNA for Z (M_z) has matching distributions, because the other two mRNA variables (M_x and M_y) are included in the ODE system. Therefore, we can see from Figure 6.4 that M_x and M_y exhibit much

smaller variances than they do in the full stochastic simulation. We do not show comparisons for strategies C and D, because they give the same results as strategy B (Figure 6.5).

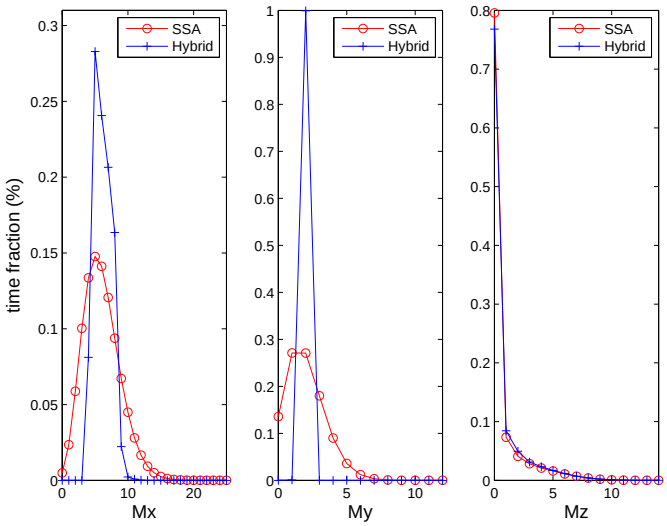


Figure 6.4: mRNA distributions by the hybrid method and the SSA on Kar et al.'s cell cycle model (Partitioning strategy A)

Next we compare the overall performance of the hybrid method with the four strategies to the SSA on the full model. The data (Table 6.1) are collected from runs of 20,000 cycles. While the four strategies and the full SSA generate the same mean values for cycle time and division size, we observe differences in the CVs. Strategy A includes only 5 stochastic reactions, so it achieves the best efficiency, which is over 100 times faster than the full SSA simulation. On the other hand, it captures only 81% and 93% of the intrinsic noise in the cycle time and division size distributions, respectively. Under conditions where computational cost is our major concern, it can still serve as a useful partitioning strategy. Strategy B puts all (and only) the gene expression reactions into the SSA regime. It captures over 97% of the noise

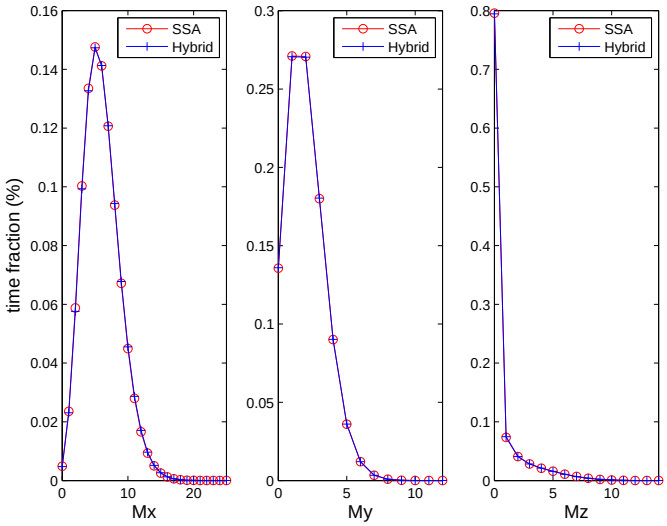


Figure 6.5: mRNA distributions by the hybrid method and the SSA on Kar et al.'s cell cycle model (Partitioning strategy B)

in the system and is 5 times faster than SSA, but it takes 22 times longer than strategy A. Considering the tradeoff between accuracy and efficiency, this strategy appears to be the best among the four. Both strategy C and D include more reactions in the stochastic system, but they do not gain thereby much improvement in simulating stochastic effects of the system. Since strategies C and D are considerably less efficient computationally than strategy B, there is little to commend them.

6.3.4 A brief analysis of the Partitioning Strategy

The numerical experiments of Kar et al.'s model suggest that, with appropriate threshold values, the hybrid method with the new partitioning strategy, which puts only reactions in

Strategy	Cell cycle time		Volume at division		CPU time(s)
	Mean(min)	CV(%)	Mean(fL)	CV(%)	
A	115.5	10.4	30.3	7.5	386
B	115.5	12.6	29.2	8.1	8,613
C	115.5	12.9	29.2	8.2	14,474
D	115.5	12.5	29.2	7.9	37,011
Full stochastic	115.5	12.9	29.1	8.1	41,774

Table 6.1: Statistics by different partitioning strategies of the hybrid method and the full Gillespie SSA on Kar's cell cycle model

region I into the SSA regime, provides an accurate and efficient stochastic simulation tool for this cell cycle model. But this may not be generally true for all types of models. We need to be careful on when we can/cannot apply this new partitioning strategy.

To further study this new partitioning strategy, we adopt a Poisson process formulation of the SSA proposed by Anderson [75]. Let $k_j(t)$ denote the total number of firing of reaction R_j from initial time 0 to t. Then

$$x(t) = x(0) + \sum_{j=1}^{M} v_j k_j(t). \tag{6.6}$$

Anderson [75] showed that $k_j(t)$ can be formulated as

$$k_j(t) = Y_j \left(\int_0^t a_j(x(s))ds \right), \tag{6.7}$$

where all Y_j are independent unit-rate Poisson processes. Anderson called the integral $I_j(t) = \int_0^t a_j(x(s))ds$ the internal time, which determines the intensity inside a unit-rate Poisson

process. I_j is also the time integral of the propensity function and determines the scale of the reaction R_j. Combining (6.6) and (6.7) we have

$$x(t) = x(0) + \sum_{j=1}^{M} v_j Y_j \left(I(t) \right). \tag{6.8}$$

Note that if we take the mean value for Y_j, we will end up with

$$x(t) = x(0) + \sum_{j=1}^{M} v_j \left(\int_0^t a_j(x(s))ds \right), \tag{6.9}$$

which is the integral format of the reaction rate equations. If we know the state at time t and would like to consider the state change from time t to $t+h$, we have

$$x(t+h) = x(t) + \sum_{j=1}^{M} v_j P_j \left(\int_t^{t+h} a_j(x(s))ds \right), \tag{6.10}$$

where all $P_j(\lambda)$'s are independent Poisson random numbers with mean and variance equal to λ. Note that although (6.10) looks similar to the tau-leaping method proposed by Gillespie[47], it is an exact representation while tau-leaping is an approximation.

Now consider different situations for a reaction R_j. If R_j is in Region I or IV, the situation is relatively easy. In region I, at least one of the reactants/products is of small population and the reaction is slow. We should put this reaction into the SSA regime to exactly simulate its firing. In region IV, all reactants and products are of large populations and the reaction is fast. According to Haseltine and Rawlings[22], this reaction can be put into the ODE or the CLE regime. We can also derive this from equation (6.10). During a time interval $(t,\ t+h)$ between two region I reaction firings, we calculate the state change of x. When R_j is in region IV, $\int_t^{t+h} a_j(x(s))ds \gg 1$, the Poisson random number can be approximated by

a Gaussian random number. This justifies the CLE representation. When $\int_t^{t+h} a_j(x(s))ds$ is large enough so that its square root is also much greater than 1, the standard deviation of the Poisson random number is negligible compared to the mean value. We can simply use its mean value, which leads to the ODE representation for R_j. Note that there are still numerical errors because either the ODE representation or the CLE representation is just an approximation. But if all the involved species are of large populations, the errors caused by the approximation are relatively small and will not affect the system behavior significantly. If R_j is in region II, all reactants and products of R_j are of large populations and the reaction is slow. Because the reaction is slow, $\int_t^{t+h} a_j(x(s))ds$ is small and the corresponding Poisson random number cannot be approximated by either a normal random number or its mean value. However, since all the involved species are of large populations, the previous argument is still valid. Even if we only keep the mean value and ignore the variance, the effect on the state variables is negligible. Thus it is safe to put R_j in region II into the ODE or the CLE regime[1].

If R_j is in region III, there are species with large populations in reaction R_j. For them, the situation is similar to the case when R_j is in region IV. With fast reactions and species with large populations, there is no large errors in those species if R_j is put in the ODE or CLE regime. But since R_j is in region III, at least one of the reactants/products is of small population. Errors caused by an ODE or CLE approximation are relatively large in this

[1]Depending on the scale difference, one may want to put R_j in the tau-leaping regime. But the implementation details will have to change accordingly. For simplicity we only discuss using ODEs to simulate the fast reactions.

case. Here we do not aim to find general conditions such that R_j can be put into the ODE regime, instead we focus on two conditions that suit our cell cycle model.

Condition 1: *For all species with small populations and whose values get changed by any reaction in region III, if they are involved in a reaction in any region, the propensity of this reaction is linear with the state variables of these (small) species.*

Condition 2: *The sum of propensities for all reactions in region I is much smaller than the propensity of any reaction in region III.*

Remark: Condition 1 requires that there is no reaction among small species, while condition 2 requires that the total firing numbers in region I should not be comparable to the reacting number of any region II reaction. Condition 2 is a necessary efficiency requirement since the efficiency of the hybrid method depends heavily on how frequently reactions in the SSA regime fire.

Assume that these two conditions are satisfied. Let the species with a small population be S_i, whose population x_i gets changed by a reaction R_j in region III. Then $v_{ij} \neq 0$. For x_i we have

$$x_i(t) = x_i(0) + \sum_{j=1}^{M} v_{ij} Y_j \left(I_j(t) \right). \tag{6.11}$$

If $I_j(t)$ is large, $Y_j(I_j(t))$ will also be large. But since x_i is of small population, there must be at least a reaction R_l to change x_i in an opposite direction, in other words, $v_{il} v_{ij} < 0$. We can rewrite (6.11) as

$$x_i(t) = x_i(0) + \sum_{v_{ij}>0} v_{ij} Y_j \left(\int_0^t a_j(x(s)) ds \right) + \sum_{v_{ij}<0} v_{ij} Y_j \left(\int_0^t a_j(x(s)) ds \right). \tag{6.12}$$

Because of condition 2, it is impossible that reactions that change x_i in an opposite direction to R_j all come from region I. There must be at least a reaction R_l in region III such that $v_{il}v_{ij} < 0$. Thus x_i is changed by fast reactions in two directions, while it maintains a low population level. It has to be at a quasi-steady-state [19, 20, 21]. According to the analysis by Rao and Arkin[19], if x_i is involved only in reactions whose propensities are *linear* with the small state variables, we only need its mean value to calculate the propensities of slow reactions, and its mean value can be solved from ODEs formed by those reactions in region III.

To see a counter example when condition 1 is broken, we consider a simple system

$$\begin{aligned} X_1 &\underset{c_2}{\overset{c_1}{\rightleftharpoons}} X_2, \quad (c_1 = c_2 = 10,000), \\ X_1 + X_2 &\xrightarrow{c_3} X_3, \quad (c_3 = 1). \end{aligned} \tag{6.13}$$

If both X_1 and X_2 are of small populations, or in the extreme case, $x_1 + x_2 = 1$, then no X_3 will be produced since there is no way there could be an X_1 molecule and an X_2 molecule bind together to form an X_3 molecule. However, if we solve the pair of fast reversible equations in ODEs, we will have $x_1(0.5) = x_2(\infty) = 0.5$ and there will be a positive propensity for the slow reaction. Of course that will lead to big errors.

If condition 2 is broken, we could have a system as follows:

$$\begin{aligned} X_0 &\longrightarrow \emptyset, \quad \text{with rate } 2,000. \\ X_i &\longrightarrow X_0 + X_i, \quad c_i = 1,\ x_i = 1, \text{ but } i = 1, \ldots, 10,000. \end{aligned} \tag{6.14}$$

The first reaction consumes X_0 quickly, while the other $10,000$ reactions generate X_0 slowly. The first reaction is in region III and the other $10,000$ reactions are in region I. We can see

that for this system, x_0 will be around five at a steady state. When we apply the hybrid method by solving the first reaction in ODEs and the rest in the SSA regime, the accuracy may be fine but the ODE solver will be interrupted by SSA events so frequently that the efficiency will be even lower than solving the whole system by the SSA.

The good news is, conditions 1 and 2 are often satisfied in gene regulation networks, where proteins are of large populations, genes and mRNAs are of small populations, and reactions in region III are regulation reactions on genes and mRNAs by proteins. Of course, the accuracy and efficiency of the hybrid method depend on the threshold values ($\bar{x}$ and $\bar{a}$ in Figure 6.2) and actual scale differences in a problem. In general, it remains an open question how to select these threshold values. For Kar et al.'s model, the threshold values are determined so that all (and only) gene expression reactions are in the SSA regime. This seems to be a natural choice for gene regulation models where proteins, genes, and mRNAs demonstrate clear scale differences. To further test how this strategy works for this type of systems, we did numerical experiments with two gene regulation models. One generates a steady state and the other generates an oscillation at the protein level. Details of these two models are given in the appendix. As shown by the experimental results, similar to what we have seen for Kar et al.'s model, the hybrid method with this partitioning strategy generates system dynamics and state statistics reasonably close to the SSA results for these two models as well.

6.3.5 Test the Partitioning Strategy with Two Gene Regulation Models

To test the accuracy of the hybrid method with the new partitioning strategy, we apply it to two gene regulation models. Their diagrams are shown in Figure 6.6.

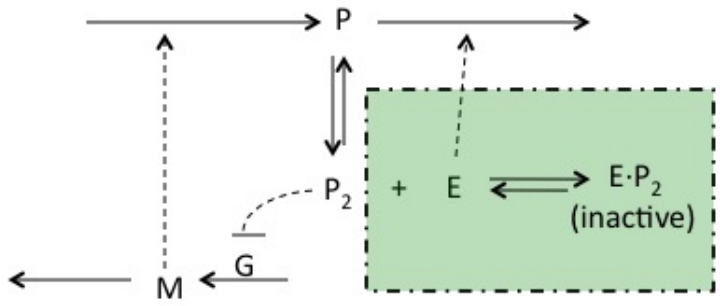


Figure 6.6: Diagrams for the two test models. The steady state model does not include reactions in the shaded box, while the oscillation model includes them.

The first test case is a negative feedback model. The reactions are given in Table 6.2. In this model, a protein p regulates its own expression by forming a homodimer ($p2$), which binds to the promotor site of its own gene and inactivates gene expression. The average populations are set as typical values in cells. The reaction rates are chosen so that the model contains reactions in four regions.

We simulate the system with three methods: the SSA, the hybrid method with only reactions in region I in the SSA regime, and the hybrid method with only reactions in region IV in the ODE regime. Note that for all methods, the distributions of the mRNAs are very close (Figure 6.7). There are noticeable differences for the distributions of the protein p (Figure

Reaction	Rate constant	Average # molecules	Average propensity	Reaction region
$g \rightarrow g + m$	2	10	1	I
$m \rightarrow m + p$	0.1	1000	1	II
$p \rightarrow \phi$	0.001	1000	1	II
$m \rightarrow \phi$	0.1	10	1	I
$p + p \rightarrow p2$	0.001	100	1000	IV
$p2 \rightarrow p + p$	10	100	1000	IV
$p2 + g \rightarrow gi$	20	0.5	1000	III
$gi \rightarrow p2 + g$	2000	0.5	1000	III

Table 6.2: Reactions in the steady state model

6.8) and its dimer $p2$ (Figure 6.9). But the differences are actually typical for species with medium populations simulated by the hybrid method. We can see much higher differences for $p2$ as its population number is 10 times smaller than that of p. Note that the mean values for both p and $p2$ are quite close to the SSA results.

In many cases, the total protein level is more important as that is what is actually measured in wet-lab experiments. If we compare the distributions of the total protein level $pTotal = p + 2 * p2$, they are much closer (Figure 6.10).

Of course, the accuracy should also depend on the scale gaps between region I and other regions. We scaled the total population of p up and down by ten-fold and observed the distributions of $pTotal$ by the SSA and the hybrid method with the new partitioning strategy

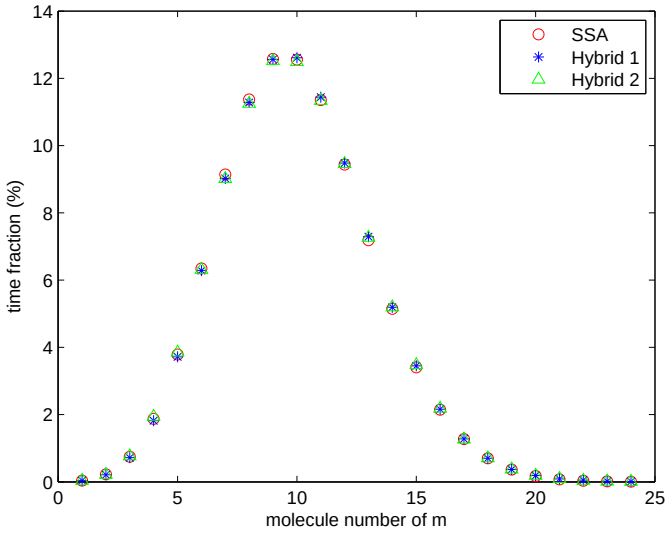


Figure 6.7: Distribution of mRNA by the steady state model

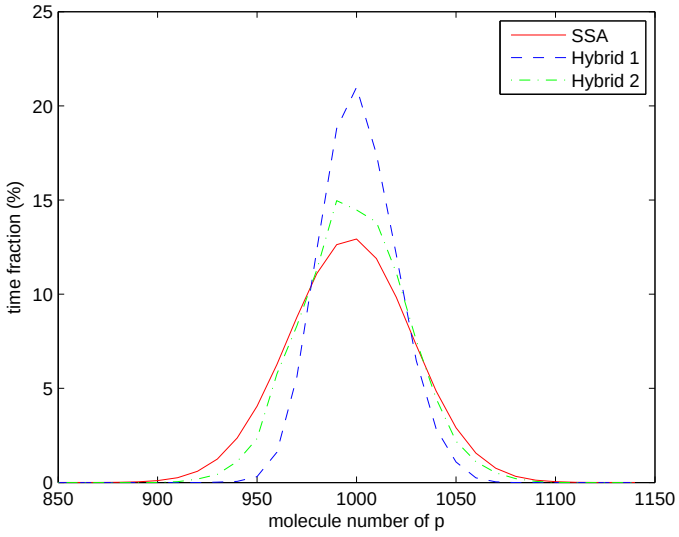


Figure 6.8: Distribution of p by the steady state model

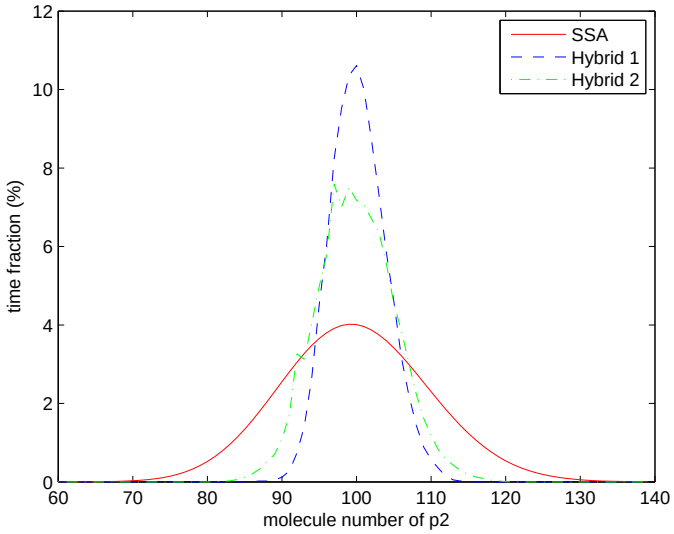


Figure 6.9: Distribution of $p2$ by the steady state model

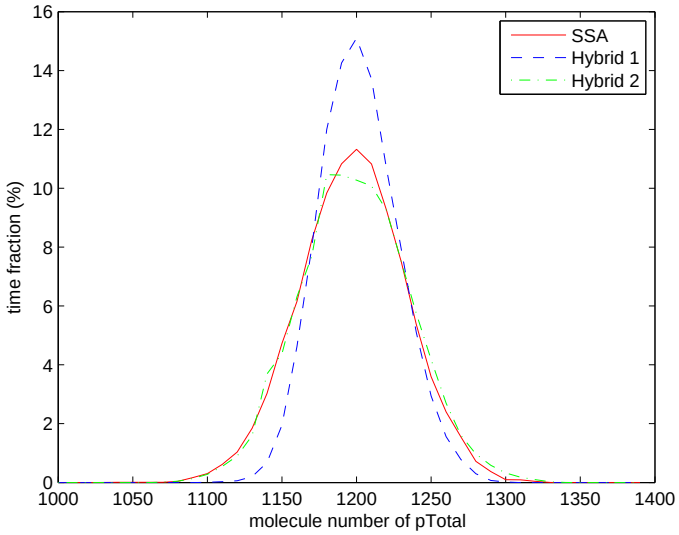


Figure 6.10: Distribution of $pTotal$ by the steady state model

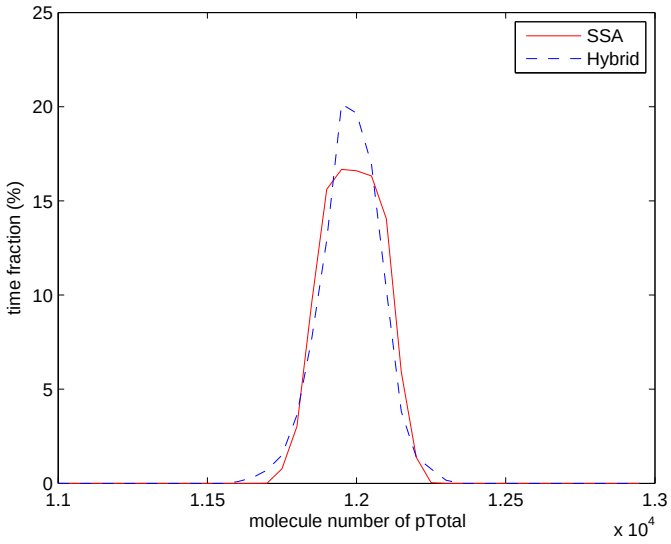


Figure 6.11: Distribution of $pTotal$ by the steady state model with 10 times larger total population p

(Figure 6.11 and 6.12). We can see similar patterns in the distribution differences, while the mean values are still quite close. The important difference is on the relative errors. For a larger population of $pTotal$, even though there is still a distribution difference between the SSA and the hybrid method, the relative error is smaller, while for a smaller population of $pTotal$, the relative error is larger. A larger relative error may have greater impact on a system property.

To determine how errors in protein level affect the accuracy of interesting system behaviors, we examined a second test case, an oscillation model. This model is an extension to the steady state model. In this model, as before, protein p forms a homodimer, $p2$, which binds to the promotor region of the gene encoding p and inhibits the gene's expression. In addition, $p2$ binds to and inhibits the enzyme E that catalyzes the degradation of protein p. For suitable

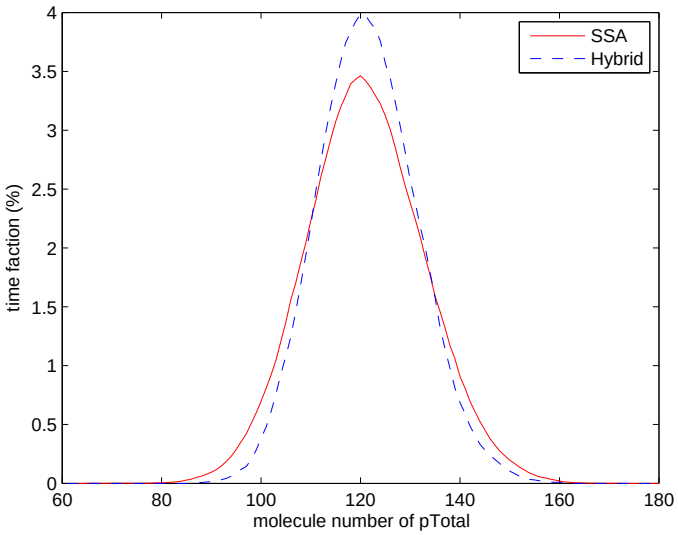


Figure 6.12: Distribution of $pTotal$ by the steady state model with 10 times smaller total population p

choice of parameter values, this system exhibits spontaneous limit cycle oscillations [76](see Figure 6.13 and 6.14). The diagram is given in Figure 6.6 and reactions are given in Table 6.3. In the partitioning, species E has a relatively small population and the pair of reactions between E and $p2$ are relatively slow. But they are still put into the ODE regime, and we expect some errors for that. The important characteristic for this system is the period of the oscillation. The statistics of the period by different simulation methods are shown in Table 6.4. We can see that the hybrid method with the new partitioning strategy works very well for this model. With the original partitioning strategy, the accuracy of the hybrid method is almost the same as with the new partitioning strategy, but the CPU time is even longer than the SSA. That is usually because of frequent firings for reactions in region III.

Reaction	Rate constant	Average # molecules	Average propensity	Reaction region
$g \rightarrow g + m$	0.5	6.3	0.3	I
$m \rightarrow m + p$	60	1439	375	IV
$p \rightarrow \phi$	0.05	1439	72	IV
$m \rightarrow \phi$	0.05	6.3	0.3	I
$p + p \rightarrow p2$	0.001	120	4799	IV
$p2 \rightarrow p + p$	40	120	4780	IV
$p2 + g \rightarrow gi$	40	0.4	459	III
$gi \rightarrow p2 + g$	1000	0.4	402	III
$E + p2 \rightarrow Ep2$	0.4	23	22	IV
$Ep2 \rightarrow E + p2$	0.5	23	23	IV
$E + p \rightarrow Ep$	0.2	23	602	IV
$Ep \rightarrow E + p$	10	23	300	IV
$Ep \rightarrow E$	10	23	300	IV

Table 6.3: Reactions of the oscillation model

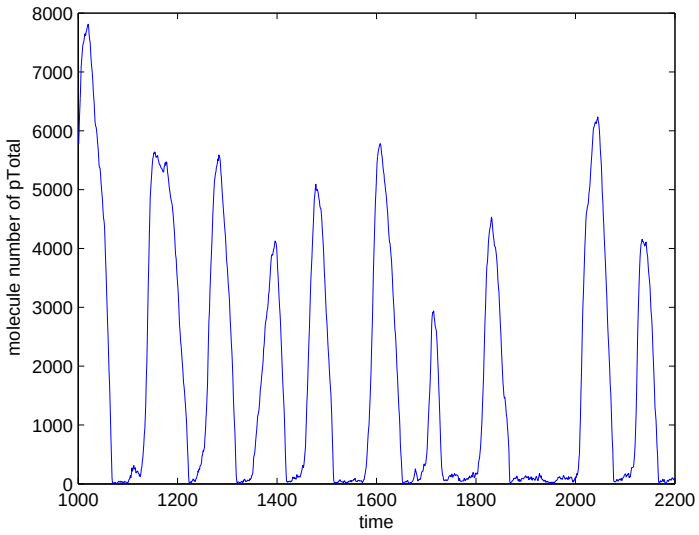


Figure 6.13: Trajectory of pTotal in the oscillation model

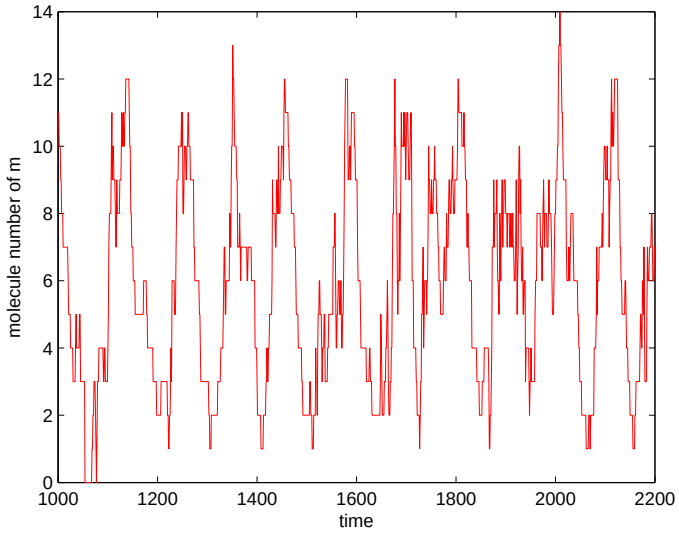


Figure 6.14: Trajectory of mRNA in the oscillation model

Simulation Methods	Mean	CV	CPU time(s)
SSA	123	37%	37
Hybrid method with the new partitioning strategy	126	41%	0.8
Hybrid method with the original partitioning strategy	126	40%	40

Table 6.4: Statistics of the period of oscillation for different methods simulating the stochastic oscillation model

6.4 A Hybrid Cell Cycle Model

The hybrid simulation method demonstrates good performance on Kar et al.'s cell cycle model. However, the whole process still requires a modeler to build a complex, full-stochastic model by converting the phenomenological rate laws into many stochastic elementary reactions. Moreover, after the conversion, the simulator has to appropriately partition the system into the SSA regime and the ODE regime with a good partitioning strategy. We have already noticed that the best partition strategy on Kar et al.'s model is to put all gene and mRNA reactions into the SSA regime. All of these reactions were absent from the three-variable model, and were added to it later to account for stochastic effects of transcription-translation coupling [70, 71, 72]. If (as we have shown for Kar et al.'s model) the gene and mRNA reactions are primarily responsible for intrinsic noise, it may not be necessary to unpack the original deterministic model for the protein regulatory network. One only needs to apply the hybrid method on a naturally partitioned model, where the SSA regime includes all newly added stochastic reactions at the gene expression level, while the ODE regime includes the

ODE set from the original deterministic model at the protein level. This is an efficient way to model stochastic gene regulation systems.

To demonstrate this modeling strategy, we propose a new "hybrid cell cycle model".

ODE system. In the deterministic part, we inherit most of the original ODE system from the three-variable model, but modify it in the following ways, to match with Kar et al.'s model.

1. In the deterministic model, a fourth variable Y_T and its corresponding ODE are added to represent the dynamic of the total amount of unphosphorylated and phosphorylated Y.

2. Originally the activation of Z by X is modeled by a Hill function with $n = 4$. In Kar et al.'s model, this process was unpacked and modeled by dimerization of a phosphorylated transcription factor. To match Kar et al.'s model, we used $n = 2$ in our hybrid model. Because this activation is accomplished at the gene expression level, we moved this nonlinear term from the ODE for Z into the stochastic system.

3. To improve the cell cycle oscillation so that it is more precise and robust, we set the basal rate for the dephosphorylation of Y (k_{hy}) to be non-zero, so that the reaction has non-zero rate even when $Z = 0$.

4. In the three-variable model, all variables are in concentrations. To convert the model from concentration-based to molecule-number-based, we changed the ODEs and pa-

rameter values accordingly.

SSA system. In the stochastic system, we introduce six stochastic reactions for the synthesis and degradation of the three mRNA variables, namely M_x for X, M_y for Y_T and Y, and M_z for Z. All of these reactions use mass-action rate laws, except the synthesis of M_z, whose propensity function includes the nonlinear term for activation of Z.

With these changes in mind, we slightly tune the parameters of some reactions in our system to have similar cell cycle behaviors and statistics as in Kar et al.'s model. The details of the hybrid model are listed in Tables 6.5, 6.6 and 6.7.

$$\frac{\mathrm{d}}{\mathrm{dt}}V = \mu V$$

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle X\rangle = k_{\mathrm{sx}}M_{\mathrm{x}}V - k_{\mathrm{dx}}\langle X\rangle - \frac{k_{\mathrm{dxy}}\langle Y\rangle\langle X\rangle}{V}$$

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle Y_{\mathrm{T}}\rangle = k_{\mathrm{sy}}M_{\mathrm{y}}V - k_{\mathrm{dy}}\langle Y_{\mathrm{T}}\rangle$$

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle Y\rangle = k_{\mathrm{sy}}M_{\mathrm{y}}V - k_{\mathrm{dy}}\langle Y\rangle + \frac{(k_{\mathrm{hy}}V + k_{\mathrm{hyz}}\langle Z\rangle)(\langle Y_{\mathrm{T}}\rangle - \langle Y\rangle)}{J_{\mathrm{hy}}V + \langle Y_{\mathrm{T}}\rangle - \langle Y\rangle} - \frac{k_{\mathrm{pyx}}\langle X\rangle\langle Y\rangle}{J_{\mathrm{pyx}}V + \langle Y\rangle}$$

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle Z\rangle = k_{\mathrm{sz}}M_{\mathrm{z}}V - k_{\mathrm{dz}}\langle Z\rangle$$

Table 6.5: ODE system of hybrid cell cycle model. $\langle X\rangle$ denotes the average number of molecules of species X.

The hybrid cell cycle model can be naturally simulated by the hybrid method. The results are compared to the full Gillespie simulation on Kar et al.'s model (Figure 6.16 and Table 6.8). The hybrid model exhibits oscillations of the three proteins of X, Y and Z (Figure 6.15) that are comparable to Kar et al.'s SSA results. The mRNA distributions of the hybrid method also agree quite well with Kar et al.'s calculation (Figure 6.16).

Reaction	Propensity function
$\phi \rightarrow \mathrm{M_x}$	$k_{\mathrm{smx}} V$
$\mathrm{M_x} \rightarrow \phi$	$k_{\mathrm{dmx}} M_{\mathrm{x}}$
$\phi \rightarrow \mathrm{M_y}$	k_{smy}
$\mathrm{M_y} \rightarrow \phi$	$k_{\mathrm{dmy}} M_{\mathrm{y}}$
$\phi \rightarrow \mathrm{M_z}$	$k_{\mathrm{smz}} + \frac{k_{\mathrm{smzx}} \langle X \rangle^2}{(J_{\mathrm{smzx}} V)^2 + \langle X \rangle^2}$
$\mathrm{M_z} \rightarrow \phi$	$k_{\mathrm{dmz}} M_{\mathrm{z}}$

Table 6.6: SSA system of hybrid cell cycle model. $\langle X \rangle$ denotes the average number of molecules of species X.

Table 6.8 shows that the statistics generated by these two models agree reasonably well with each other and with the experimental data. Kar et al.'s model was parameterized to give a nominal size at division of 30 fL, without much regard for the actual size of yeast cells at division. This discrepancy can be corrected by adjusting the rate constants in Table 6.7. To increase V by a factor of F, k_{dxy} must be multiplied by F and all parameters with units fL^{-1} or $\mathrm{fL}^{-1} \cdot \mathrm{min}^{-1}$ must be divided by F.

Our hybrid model produces larger CVs of the cycle time and division size than Kar et al.'s model. Similar to Kar et al.'s model, our hybrid model includes low numbers of mRNAs. The average levels of $\mathrm{M_x}$, $\mathrm{M_y}$ and $\mathrm{M_z}$ are respectively 6.4, 2.0, and 0.5. With such low abundances, both models require short half-lives of the mRNAs in order to keep the intrinsic noise at an acceptable level. Therefore, we reuse the degradation rates of the mRNAs from Kar et al.'s model. The resulting half-lives of $\mathrm{M_x}$, $\mathrm{M_y}$ and $\mathrm{M_z}$ are respectively 0.2, 0.2, and

Parameter	Value (min^{-1})	Parameter	Value ($\text{fL}^{-1} \cdot \text{min}^{-1}$)
μ	0.006	k_{sx}	1.53
k_{dx}	0.04	k_{sy}	1.35
k_{dy}	0.02	k_{hy}	29.7
k_{hyz}	7.5	k_{sz}	1.35
k_{pyx}	1.88	k_{smx}	1.04
k_{dz}	0.1	Parameter	Value ($\text{fL} \cdot \text{min}^{-1}$)
k_{dmx}	3.5	k_{dxy}	0.00741
k_{smy}	7.0	Parameter	Value (fL^{-1})
k_{dmy}	3.5	J_{hy}	5.4
k_{smz}	0.001	J_{pyx}	5.4
k_{smzx}	10.0	J_{smzx}	756
k_{dmz}	0.15		

Table 6.7: Parameter values of hybrid cell cycle model

4.6 min, the same as in Kar et al's model. The hybrid model reduced the simulation time by a factor of 40. Moreover, from the modeling point of view, we avoid great difficulties that come from developing a complex chemical network, as in Kar et al.'s work.

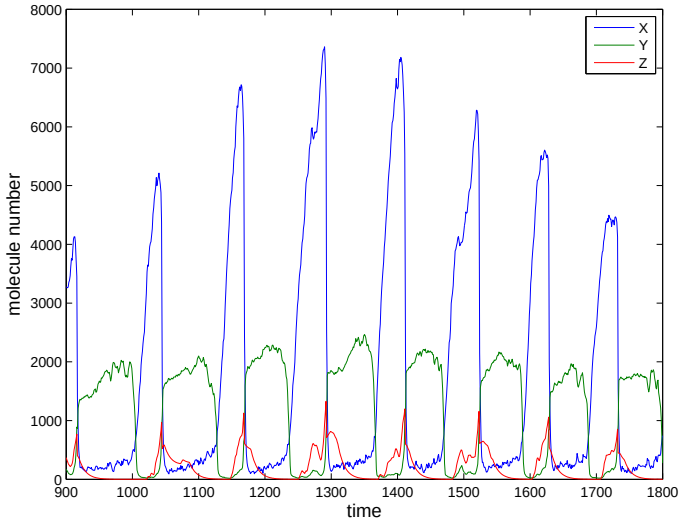


Figure 6.15: Time trajectory of the hybrid model

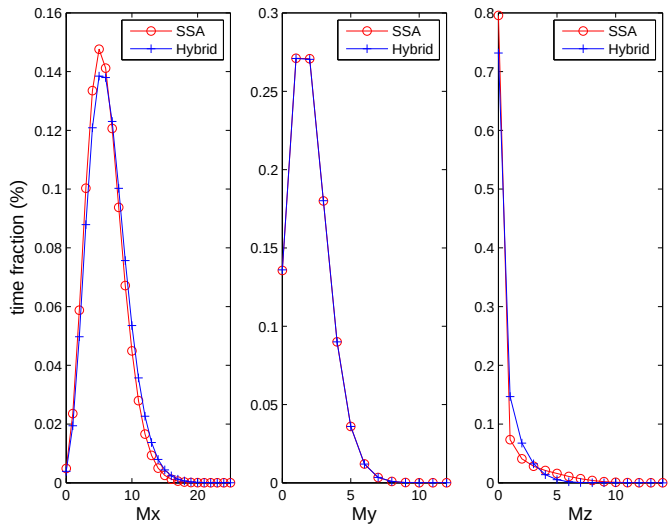


Figure 6.16: mRNA distributions of the hybrid model with hybrid simulation and Kar's model with full SSA simulation

	Cell cycle time		Volume at division		CPU time(s)
	Mean(min)	CV(%)	Mean(fL)	CV(%)	
Fission yeast	116	14	175	8	NA
Budding yeast (daughter)	112	22	68	19	NA
Kar's	116	13	29	8	41,774
Hybrid	116	20	30	12	1,370

Table 6.8: Statistics of cell cycle models. Row 1 includes experimental data from a fission yeast cell sample [1]. Row 2 includes the experimental data for daughter cells of the budding yeast [2]. Rows 3 and 4 are statistics respectively by the SSA on Kar's model and the hybrid method on the hybrid model. NA: Not Applicable.

Chapter 7

Stochastic Simulation on Models Based on Multiple Site Phosphorylation

7.1 Motivation

As we have mentioned in the previous chapter, to build accurate stochastic biochemical models, one big challenge is the selection of the rate law equations for those non-elementary chemical reactions that form the positive and negative feedback loops. These feedback loops are usually indispensable for the regulation of the cell cycle network, because they are often supposed to create a certain level of "nonlinearity" and maintain bistable switch behaviors.

Besides breaking down simple switch systems into elementary reactions with mass-action rate laws as Kar et al. did [11], another method was discovered by Qu et al. [56]. This discovery shows that modeling the regulative protein with multiple site phosphorylation may also produce the "nonlinearity" in the system and generate bistable switch behavior. The idea is to assume that the regulative protein have multiple phosphorylation levels, which enable it to undergo multiple (de)phosphorylation reactions between different levels. Figure 1.4 shows an example of a bistable switch based on this idea. It manages to achieve the required nonlinearity with only mass-action kinetics at the cost of a relatively more complicated network. However, this modeling technique potentially presents challenges for the current simulation algorithms, because models based on multiple site phosphorylation may possess both the multi-state and multi-scale features.

In this chapter, we first study the multi-scale feature of the bistable switch model involving multiple phosphorylation. Although the literature provides us a handful of available approximation methods to speed up the SSA, there is no convenient criterion on whether or not we should select a particular approximation method. To do so there has to be a solid numerical analysis to access the accuracy and efficiency of these methods. We focus on a simple model motivated by the cell cycle model based on multiple site phosphorylation. We will study such a criterion to evaluate the performance with the multi-scale feature in the system. By efficiently estimating the relaxation time of the subsystem in the fast time scale, we are able to analyze the performance of the SQSSA and Haseltine and Rawlings' hybrid method. Secondly, we focus on the multi-state feature of a budding yeast cell cycle model which is

based on multiple site phosphorylation. To accelerate the simulation, the cell cycle model is formulated as a rule-based model and simulated with the rule-based simulation methods proposed in Chapter 5. Numerical results are presented for both multi-scale and multi-state simulations.

7.2 Background

7.2.1 SQSSA/ssSSA

The SQSSA[19] and ssSSA [20, 21] were proposed to reduce the size of a stochastic biological model and to improve the stochastic simulation efficiency. The ssSSA is based on the partial equilibrium (PE) assumption, while the SQSSA is based on the quasi steady state (QSS) assumption. The PE and QSS assumptions are both important multiscale features in biochemical systems. PE refers to the situation where some reactions fire much faster than others and the corresponding subsystem reaches a partial equilibrium state[20]. QSS refers to the situation where some state variables fluctuate very quickly around their quasi steady states[19]. If a subsystem is at PE, then all its involved species are in QSS. For this reason, in this paper we do not clearly distinguish these two methods and will just refer both of them the SQSSA method.

To implement the SQSSA method, a system is partitioned into fast and slow subsystems, and the state variables are partitioned into fast and slow variables. The fast subsystem is

assumed at an equilibrium state, while the state variables in the fast subsystem are in QSS. Then the simulation procedure is very similar to the original SSA procedure, except that in each simulation step, a set of algebraic equations is solved to find the steady state of all fast variables. Then these fast variables are considered (temporary) constant parameters in the slow subsystem and the SSA procedure is applied only to the slow subsystem. Because the number of reaction firings in the slow subsystem is much less than the one for the fast subsystem, high simulation efficiency is achieved. However, if the time scale difference is not large enough, the QSSA method may lead to large errors.

7.3 Relaxation Time Criterion for SQSSA/ssSSA

7.3.1 Application of SQSSA on Multiple Site Phosphorylation Model

SQSSA assumes that certain species reach stochastic steady states rapidly with frequent firings of fast reactions. However, the definitions of the fast and slow reactions are ambiguous and depend on the choice of the user. In general, fast reactions are supposed to have large propensities along the entire simulation, with high reaction rates or large reactant populations. These reactions usually fire much more frequently than the others, but once the involved species are at steady states these reactions just cancel each other and the overall system dynamics is driven by firings of slow reactions. The SQSSA simplifies the simula-

tion process by replacing the intensive simulation of fast reactions with a set of algebraic equations. This technique can be applied to the bistable switch of the cell cycle model as described in (1.4), if we can assume that the phosphorylation and dephosphorylation reactions are fast enough. We can naturally suppose the whole species of $Cdh1$ under different phosphorylation levels reach steady states after a short time.

For purpose of simplicity, let's first suppose $Cdh1$ has only three phosphorylation levels instead of eleven levels in the real model. We will see later that our result does not depend on the number of phosphorylation levels. Thus, we currently focus on the following chain reactions

$$S_0 \underset{c_3}{\overset{c_1}{\rightleftharpoons}} S_1 \underset{c_4}{\overset{c_2}{\rightleftharpoons}} S_2, \tag{7.1}$$

where c_1 through c_4 are rate constants for corresponding reactions. Note that we denote the three levels of $Cdh1$ by S_0, S_1 and S_2 for easier representation. To make the analysis even simpler, we suppose that there is only one molecule of S in total. Let the state vector be represented in the order of S_0, S_1 and S_2. Then there are only three possible states for this simple chain system: $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ with probabilities $P_0(t)$, $P_1(t)$ and $P_2(t)$, e.g. state $(1,0,0)$ means the molecule is an S_0 with probability P_0 at time t. We can thus derive the following set of ODEs from the Chemical Master Equations (CMEs):

$$\begin{aligned} P_0' &= c_3 \cdot P_1 - c_1 \cdot P_0, \\ P_1' &= c_1 \cdot P_0 + c_4 \cdot P_2 - (c_2 + c_3) \cdot P_1, \\ P_2' &= c_2 \cdot P_1 - c_4 \cdot P_2. \end{aligned} \tag{7.2}$$

Applying the SQSSA, we assume after an infinitesimal transient period the system reaches stochastic equilibrium, which means the probabilities remain unchanged, or in other words, all derivatives in (7.2) become zero. Let P_0^*, P_1^* and P_2^* be the values of P_0, P_1 and P_2 at the stochastic equilibrium state. We have

$$\begin{aligned} c_3 \cdot P_1^* - c_1 \cdot P_0^* &= 0 \\ c_1 \cdot P_0^* + c_4 \cdot P_2^* - (c_2 + c_3) \cdot P_1^* &= 0 \\ c_2 \cdot P_1^* - c_4 \cdot P_2^* &= 0. \end{aligned} \tag{7.3}$$

The equations in (7.3) are not independent. We need another equation to solve for the steady state probabilites. Note that P_0, P_1, and P_2 cover all possibilities of species S, we have

$$P_0^* + P_1^* + P_2^* = 1. \tag{7.4}$$

Then we can solve (7.3) and (7.4) and get

$$\begin{aligned} P_0^* &= \frac{c_3 \cdot c_4}{c_1 \cdot c_2 + c_1 \cdot c_4 + c_3 \cdot c_4} \\ P_1^* &= \frac{c_1 \cdot c_4}{c_1 \cdot c_2 + c_1 \cdot c_4 + c_3 \cdot c_4} \\ P_2^* &= \frac{c_1 \cdot c_2}{c_1 \cdot c_2 + c_1 \cdot c_4 + c_3 \cdot c_4}. \end{aligned} \tag{7.5}$$

Now, suppose the chain system has a total number of N_t molecules of S, and each molecule is independent from each other. In the equilibrium state, each molecule follows the probabilities derived from (7.3) and (7.4). If we consider the state of each molecule as one independent trial, this resembles a Bernoulli trial except that it has three possible outcomes rather than two. Then we have N_t independent trials with the same probability distribution. Therefore,

the populations of the three levels follow a multinomial distribution $M(N_t, P_0^*, P_1^*, P_2^*)$. In this way, we obtain sufficient information about the whole species of S by simply solving a set of algebraic equations. The procedure for this three level case can be easily extended to models with more levels.

7.3.2 Relaxation Time

However, before applying the SQSSA in the above way, it is important to ask, how do we know the assumptions that we make are valid? To be specific, how can we ensure that the chain reactions reach equilibrium fast enough? To answer this question, we need to analyze the relaxation time[1] of the chain reactions. After a slow reaction fires, the steady state in the fast subsystem has to be changed. The relaxation time concerns the time scale for the fast system to "relax" back to another steady state. It also can be viewed as the time scale after which the fast subsystem will "forget" a previous perturbation. Thus the relaxation time in a stochastic process can be defined as the time interval after which the state of a system is no longer correlated to the current state. With this definition, it is a relatively difficult job to derive an analytic expression for the relaxation time. However, it can always be measured by calculating the correlation coefficient between the current state and the state after some time. In this paper, we will not analytically solve the correlation coefficient, but will derive an approximation of the relaxation time.

[1]Here we mix the relaxation time and the de-correlation time as in practice they are in the same order of magnitude.

Again, to make the presentation simple, we suppose there is only one molecule for S in the system (7.1). Let $P(t) = [P_0(t), P_1(t), P_2(t)]^T$, then (7.2) can be rewritten in the matrix representation as

$$P' = A_3 P, \tag{7.6}$$

where

$$A_3 = \begin{bmatrix} -c_1 & c_3 & 0 \\ c_1 & -c_2 - c_3 & c_4 \\ 0 & c_2 & -c_4 \end{bmatrix}. \tag{7.7}$$

Note that the system is closed. We always have the conservation law

$$P_0 + P_1 + P_2 = 1. \tag{7.8}$$

Thus the state vector P can be reduced to $\hat{P}(t) = [P_0(t), P_1(t)]^T$. Substituting $P_2(t)$ with (7.4), we have

$$\hat{P}' = \hat{A}_3 \hat{P} + B, \tag{7.9}$$

where

$$\hat{A}_3 = \begin{bmatrix} -c_1 & c_3 \\ c_1 - c_4 & -c_2 - c_3 - c_4 \end{bmatrix}, \tag{7.10}$$

and

$$B = \begin{bmatrix} 0 & 0 \\ 0 & c_4 \end{bmatrix}. \tag{7.11}$$

For this ODE system, the rate that each element in vector $P(t)'$ goes to zero is in the same order of the absolute value of its corresponding eigenvalue in $\hat{A}_3$, and the relaxation time can

be estimated by the maximum among the inverse of these rates. For this three level case, the eigenvalues of $\hat{A}_3$ are the solutions of λ to the following polynomial:

$$\lambda^2 + (c_1 + c_2 + c_3 + c_4)\lambda + c_1 \cdot c_2 + c_1 \cdot c_4 + c_3 \cdot c_4 = 0. \tag{7.12}$$

But this way we have to solve the equation (7.12) and select the larger value. Here we have a more efficient way to estimate the relaxation time. We use the sum, instead of the maximum, of all the inverse of these rates. This trick will make the calculation much easier. Suppose λ_1 and λ_2 are the two non-zero eigenvalues of $\hat{A}_3$ and thus are solutions of the equation (7.12). Then we have

$$\begin{aligned} \lambda_1 + \lambda_2 &= -(c_1 + c_2 + c_3 + c_4), \\ \lambda_1 \cdot \lambda_2 &= c_1 \cdot c_2 + c_1 \cdot c_4 + c_3 \cdot c_4. \end{aligned} \tag{7.13}$$

We can also prove (omitted) that λ_1 and λ_2 are both negative real number. Thus the relaxation time can be approximated as

$$\tau_{r3} = \frac{1}{|\lambda_1|} + \frac{1}{|\lambda_2|} = -\frac{\lambda_1\lambda_2}{\lambda_1 + \lambda_2} = \frac{c_1 + c_2 + c_3 + c_4}{c_1 \cdot c_2 + c_1 \cdot c_4 + c_3 \cdot c_4}. \tag{7.14}$$

For the case where the possible level is more than three, we can follow a similar way to derive a linear ODE system similar to the equation (7.9) and the maximum of the absolute values of the inverse of all eigenvalues in the coefficient matrix is the approximation of the relaxation time. But we can still play a similar trick to approximate the relaxation time by the sum of the absolute values of the inverse of all eigenvalues. Thus we have

$$\tau_{rn_l} = \sum_{i=1}^{n_l - 1} \frac{1}{|\lambda_i|}, \tag{7.15}$$

where n_l is the number of phosphorylation levels and λ_i is the i-th eigenvalue of the reduced parameter matrix $\hat{A}_{n_l}$ of size $(n_l - 1) \times (n_l - 1)$. We can prove that $\hat{A}_{n_l}$ has $n_l - 1$ negative real eigenvalues (proof omitted). Let λ_i be the i-th eigenvalue of $\hat{A}_{n_l}^{-1}$. Then we have

$$\tau_{rn_l} = \sum_{i=1}^{n_l-1} \frac{1}{|\lambda_i|} = trace(\hat{A_{n_l}}^{-1}). \tag{7.16}$$

With (7.16) we do not have to solve for all the eigenvalues of a probably large matrix, but only calculate the trace of the inverse of $\hat{A_{n_l}}$. Note that τ_r only gives an approximation of the real relaxation time of the system. But this approximation is in the same order of magnitude as the actual relaxation time. Now we can compare the approximated relaxation time with the average next reaction time of all the slow reactions. If the relaxation time is smaller, it is safe to use SQSSA. Otherwise large errors may be introduced.

7.3.3 Proof of Relaxation Time Calculation

For a set of chain reaction

$$S_0 \underset{b_1}{\overset{f_1}{\rightleftharpoons}} S_1 \underset{b_2}{\overset{f_2}{\rightleftharpoons}} \cdots \underset{b_n}{\overset{f_n}{\rightleftharpoons}} S_n, \tag{7.17}$$

where f_i's are rates for forward reactions and b_i's are rates for backward reactions, suppose there is only one molecule in total, which may take any state from S_0 to S_n. Denote $x_i(t)$ as the probability that at time t this molecule is at state S_i and $\mathbb{X}$ as the vector $\{x_0, \ldots, x_n\}$. We have

$$\sum_{i=0}^{n} x_i = 1. \tag{7.18}$$

We can also write down the chemical master equation

$$\frac{d\mathbb{X}}{dt} = A\mathbb{X}, \tag{7.19}$$

where

$$A = \begin{bmatrix} -f_1 & b_1 & 0 & \cdots & 0 \\ f_1 & -(b_1 + f_2) & -b_2 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & f_{n-1} & -(b_{n-1} + f_n) & b_n \\ 0 & \cdots & 0 & f_n & -b_n \end{bmatrix}.$$

Note that A is singular. We need to apply equation (7.18) and substitute x_0 to obtain a nonsingular matrix. From (7.18) we have

$$x_n = 1 - \sum_{i=0}^{n-1} x_i. \tag{7.20}$$

Let $\tilde{\mathbb{X}}$ denote the vector $\{x_0, \ldots, x_{n-1}\}$. We then have the equation

$$\frac{d\tilde{\mathbb{X}}}{dt} = \tilde{A}\tilde{\mathbb{X}} + y, \tag{7.21}$$

where

$$\tilde{A} = \begin{bmatrix} -f_1 & b_1 & 0 & \cdots & 0 \\ f_1 & -(b_1 + f_2) & -b_2 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & b_{(n-1)} \\ -b_n & -b_n & \cdots & f_{(n-1} - b_n & -(b_{n-1} + f_n + b_n) \end{bmatrix}.$$

and $y = (0, \ldots, 0, b_n)^T$. One can verify that the determinant of $\tilde{A}$ is

$$(-1)^n \left[\sum_{i=1}^{n-1} \prod_{j=1}^{i-1} f_j \prod_{l=i}^{n} b_l + \prod_{i=1}^{n-1} f_i (b_n + f_n) \right].$$

When all $f_i > 0$ and $b_i > 0$, this determinant is nonzero. Thus $\tilde{A}$ is nonsingular and the solution of the equilibrium equation

$$\tilde{A}\tilde{\mathbb{X}} + y = 0 \tag{7.22}$$

gives the equilibrium state for this chain reaction system. To show that this equilibrium state is stable, we need to check that all eigenvalues of $\tilde{A}$ are negative. However, $\tilde{A}$ does not have the nice structure of a tridiagonal matrix (as in A). In order to study its eigenvalues, we will still use the original matrix A, and note that the eigenvalues of A include all eigenvalues of $\tilde{A}$ and one zero. Thus all results about A's nonzero eigenvalues apply to $\tilde{A}$. We have the following theorem.

Theorem 1 All eigenvalues of A are negative real numbers except one zero.

Proof We have shown that $\tilde{A}$ is nonsingular. Thus A has one zero eigenvalue and n nonzero eigenvalues. We first note that A is a tridiagonal matrix with all off-diagonal entries positive. Thus A is similar to a symmetric matrix. Therefore all eigenvalues of A are real. Then we note that $-A$ is an M matrix, a matrix with nonpositive off-diagonal and nonnegative diagonal entries. Since all nonzero eigenvalues of an M matrix are positive[77], we have that all non-zero eigenvalues of A are negative.

With Theorem 1, we get that all eigenvalues of $\tilde{A}$ are negative real numbers. We can order them as λ_i, $i = 1, \ldots, n$ and $0 > \lambda_1 > \lambda_2 > \ldots > \lambda_n$. The relaxation time is given by $\frac{1}{|\lambda_1|}$

and we estimate it by $\sum_{i=1}^{n} \frac{1}{|\lambda_i|}$. Then we have

$$\sum_{i=1}^{n} \frac{1}{|\lambda_i|} = -\sum_{i=1}^{n} \frac{1}{\lambda_i} = -tr(\tilde{A}^{-1}). \tag{7.23}$$

Computational Cost To compute $tr(\tilde{A}^{-1})$, we need to solve for the diagonal elements in $\tilde{A}^{-1}$. In general, solving an inverse matrix of an n by n matrix takes $O(n^3)$ operations. However, $\tilde{A}$ has its special structure. In each column and in each row except the last one, there are only three nonzero elements. The computational cost is actually much lower. In the LU factorization process, each forward elimination step does not have to perform from index 1 to n, but only three rows and two columns. The computational cost for the LU factorization will take only $4n$ operations. Solving for $\tilde{A}$ after the LU factorization, to solve for all diagonal elements in $\tilde{A}^{-1}$, the computational cost will be $\frac{n^2}{2}$. Overall the computational cost for solving $tr(\tilde{A}^{-1})$ is about $\frac{n^2}{2}$, much less than $O(n^3)$.

7.3.4 The Hybrid Method

From our discussion above, when the relaxation time of the fast subsystem is much smaller than the average firing interval of the slow reactions, after any state change caused by a slow reaction firing and before the next slow reaction fires, the fast subsystem has already relaxed to its stochastic equilibrium state. Thus we conclude that it should be safe to assume the fast species remain at a stochastic equilibrium state all the time. Note that the equilibrium state itself will change when slow reactions fire. But the fast subsystem still remain at a stochastic equilibrium state.

What if the relaxation time becomes comparable to or exceeds the outside firing interval in average? In this case the firings of slow reactions can change the fast subsystem before it relaxes back to its equilibrium state. Apparently the SQSSA cannot be applied. However, even if the fast reactions are not at equilibrium, their fluctuations may still have only small impact on the whole dynamics. If we ignore their stochastic fluctuations and capture their average behavior by deterministic ODE solution, while simulate slow reactions in a discrete and stochastic manner, we can still reproduce important stochastic dynamics concerned in a biological system. This is the idea of the hybrid method. Since the deterministic simulation is usually faster, in most cases we are able to substantially improve the simulation efficiency without losing much accuracy. But by what range can the hybrid method be applied? As of now we are not aware of any theoretical analysis on the accuracy of the hybrid method except some case studies. Thus in this paper we will conduct numerical experiment with a wide of parameters to test the accuracy of the hybrid method when it is applied to the chain reaction model, which is motivated by the multiple site phosphorylation model.

7.4 Numerical Results

7.4.1 Numerical Solution of the Relaxation Time

To test the accuracy of our approximation for the relaxation time, we conduct numerical experiment on the three state chain reaction model presented in (7.1). For different groups

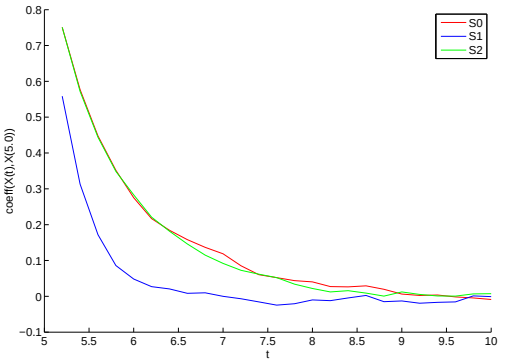
of settings for the initial populations and parameters c_1 through c_4, we compare the resulting approximated relaxation times by the method provided in Section 3 with the numerical solutions of the correlation coefficients. Table 7.1 gives the detailed descriptions of the experiments.

Experiment	*Parameters*				*Initial Populations*			τ_r
Group	c_1	c_2	c_3	c_4	S_0	S_1	S_2	
a	*1.0*	*1.0*	*1.0*	*1.0*	*333*	*334*	*333*	*1.33*
b	*10.0*	*1.0*	*1.0*	*1.0*	*48*	*476*	*476*	*0.62*
c	*1.0*	*10.0*	*1.0*	*1.0*	*83*	*83*	*834*	*1.08*

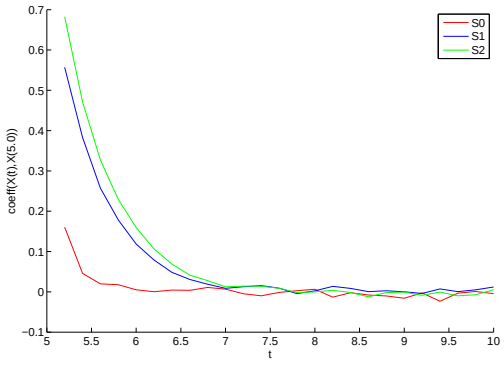
Table 7.1: Approximated relaxation times of the toy model. Parameters in three groups of experiments are tuned for the purpose of observing different relaxation times. All the initial populations are set so that the systems start near their stable states. The relaxation times are approximated by (7.14).

Figure 7.1 shows the time trajectories of the correlation coefficients for all the three species and parameter settings in Table 7.1.

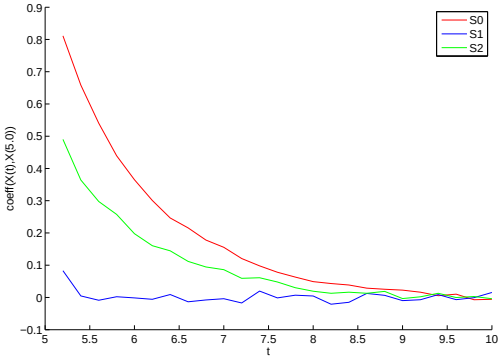
As we can see with these figures, the correlation coefficients of all three species converge to around 0 after a certain period (relaxation time). Species in experiment b become unrelated to their previous states faster, while in experiment a and c they relax slower. The numerical results match well in magnitude with the approximations in Table 7.1. Similar experiments have also been conducted on this kind of chain reaction systems with 6 and 10 levels of species S. The numerical results exhibit the same pattern as we observe for the three level model.



(a) Experiment a



(b) Experiment b



(c) Experiment c

Figure 7.1: The statistics from the each experiment are obtained from 10,000 simulation runs of the SSA. In each experiment, we first let the system proceeds for 5.0 time units so that the it is fully relaxed to its equilibrium distribution, and then start to collect numerical data by calculating each species' correlation coefficient.

7.4.2 SQSSA Application on Bistable Switch

Now, we are ready to consider the bistable switch model in (1.4). We assume the chain reactions are fast reactions and apply the SQSSA on the species $Cdh1$. It is natural that we start from the very original form of this model, where there are only two phosphorylation levels. We can formulate it as follows.

$$\begin{array}{rcl} Cdh1P_0 & \rightleftharpoons_{k\cdot k_p}^{k\cdot Clb2} & Cdh1P_1 \\ \phi & \xrightarrow{k_s} & Clb2 \\ Clb2 & \xrightarrow{k_d(Cdh1P_0)} & \phi \end{array} \tag{7.24}$$

In this model, $Clb2$ acts as an enzyme for the phosphorylation of $Cdh1$. On the other hand, the un-phosphorylated form $Cdh1P_0$ also activates the degradation of $Clb2$. Note that the degradation rate k_d largely depends on $Cdh1P_0$, as the contribution from the other form of $Cdh1$ is only a half of that from $Cdh1P_0$. The parameters are given below:

$$k_p = 8.0, k_s = 0.4, k_a = 5.5 \times 10^{-4}, \quad and$$

$$k_d = k_a(\cdot Cdh1P_0 + 0.5 \cdot Cdh1P_1).$$

Since our interest is on the accuracy of the SQSSA, we simply choose the mass-action kinetics for both the enzyme-substrate reactions. Although this setting destroys the bistable behavior due to lack of nonlinearity in the model, we are still able to compare the distributions generated by the SQSSA and the exact SSA. The validity of the SQSSA basically relies on the fact that the fast species reaches equilibrium on a higher time scale. This actually

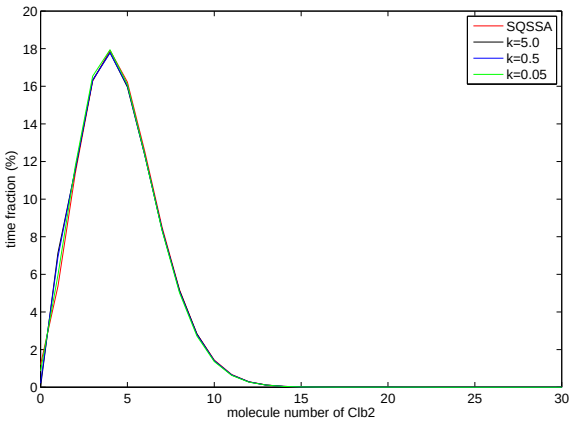


Figure 7.2: Distribution comparison between SQSSA and SSA for mono-stable model. The y-axis is the time percentage that is occupied when $Clb2$ has a molecule number as the corresponding value on the x-axis. The red curve is for the SQSSA and the others are for the SSA with different values of k. Each distribution curve is generated by one run of simulation for 5×10^6 time units.

presents an equivalent assumption for the SQSSA that $k = \infty$. So in our experiments we vary the scale parameter k and observe how the accuracy is affected.

In Figure 7.2 we see that the SQSSA generates a close distribution to all the ones by the exact SSA. Only for the curve for $k = 0.05$ can small error be observed. This result can be verified by comparing the approximated relaxation time for $Cdh1P_0$ and $Cdh1P_1$ to the average time intervals between the slow reactions. For this model, the parameter matrix is 2×2 and has only one non-zero eigenvalue which can be easily calculated, and the relaxation time is approximated by the inverse of this eigenvalue (the absolute value). This comparison is shown in Table 7.2. In the worst case where k has the smallest value 0.05, τ_r is already comparable to the slow reaction interval. However, it does not seem to affect the accuracy

very much.

k	τ_r	*Average Time Interval between Slow Reactions*
0.05	*1.59*	*1.25*
0.5	*0.159*	*1.25*
5.0	*0.0159*	*1.25*
∞	*0.0*	*1.25*

Table 7.2: Comparison between relaxation time and average time interval of slow reactions. The time interval between slow reactions is averaged as the mean value of all the intervals along one run of simulation.

Then we extend our model to a more complicated version. In a realistic cell cycle model, $Cdh1$ is usually assumed to have around ten phosphorylation levels. Following this setting, the model can be represented by the chemical reactions as follows.

$$\begin{aligned} Cdh1P_0 \underset{k_p \cdot k}{\overset{k \cdot Clb2}{\rightleftharpoons}} Cdh1P_1 \underset{k_p \cdot k}{\overset{k \cdot Clb2}{\rightleftharpoons}} \cdots \underset{k_p \cdot k}{\overset{k \cdot Clb2}{\rightleftharpoons}} Cdh1P_9 \\ \phi \xrightarrow{k_s} Clb2 \\ Clb2 \xrightarrow{k_d(Cdh1P_0)} \phi \end{aligned} \tag{7.25}$$

Here the degradation rate k_d still largely depends on $Cdh1P_0$, as the contribution from the other forms of $Cdh1$ decreases linearly as the phosphorylation level increases. As a result, both proteins form a positive feedback. With the multiple site phosphorylation and proper settings on the parameters, nonlinearity can arise even if we still choose mass-action kinetics for all the reactions. Thus, this model can exhibit bistable behaviors with the parameter sets

$$k_p = 8.0, k_s = 0.4, k_a = 5.5 \times 10^{-4}, \quad and$$

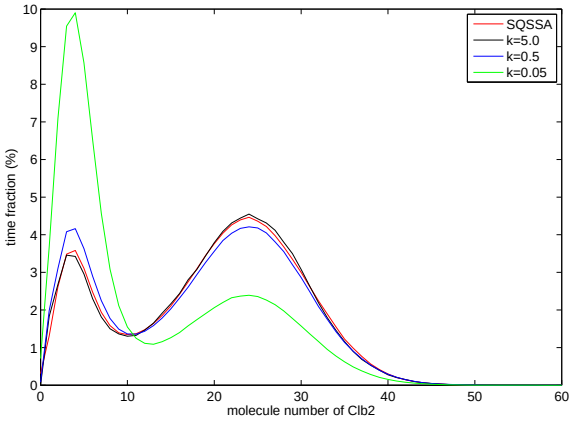


Figure 7.3: Distribution comparison between SQSSA and SSA for bi-stable model. The y-axis is the time percentage that is occupied when $Clb2$ has a molecule number as the corresponding value on the x-axis. The red curve is for the SQSSA and the others are for the SSA with different values of k. Each distribution curve is generated by one run of simulation for 5×10^6 time units.

$$k_d = \sum_{i=0}^{9} k_a \cdot (10 - i) \cdot Cdh1P_i.$$

Moreover, with more levels of phosphorylation, the relaxation time for the entire species of $Cdh1$ increases as it takes more time for ten states to reach equilibrium than it does for only two states. Table 7.3 calculates the approximated relaxation time for each choice of parameter k against the average time interval between the slow reaction firings. Figure 7.3 then shows the comparison on the bistable distributions of $Clb2$ generated by both the SQSSA and the SSA.

As we see from Table 7.3, only when $k = 5.0$ the relaxation time of the chain reactions can be considered at a higher time scale than the slow reactions. This analysis also matches very well with the statistics in Figure 7.3, which shows that the SQSSA simulation does not cause

k	τ_r	*Average Time Interval between Slow Reactions*
0.05	*24.2*	*1.25*
0.5	*1.39*	*1.25*
5.0	*0.132*	*1.25*
∞	*0.0*	*1.25*

Table 7.3: Comparison between relaxation time and average time interval of slow reactions for the extended model. The time interval between slow reactions is averaged as the mean value of all the intervals along one run of simulation. The relaxation times are approximated by (7.16).

larger errors only when $k = 5.0$, but large errors can be observed for $k = 0.5$ and $k = 0.05$.

7.4.3 Comparison between the Hybrid method and the SQSSA

As we have seen that the SQSSA requires a relatively stringent condition to be successfully applied onto the more realistic case of the bistable switch, we consider the hybrid method as an alternative and compare its accuracy with the SQSSA. As in model (7.25), the synthesis and degradation reactions are supposed to be slow reactions. So we put them into the slow subsystem for stochastic simulation. At the same time the chain reactions are put into the fast subsystem to be solved with ODEs. Then experiments are conducted separately by two configurations $k = 0.5$ and $k = 0.05$, in which cases the SQSSA has failed in maintaining good accuracy. Figure 7.4 shows the resulting distributions generated by the hybrid method comparing with the exact SSA.

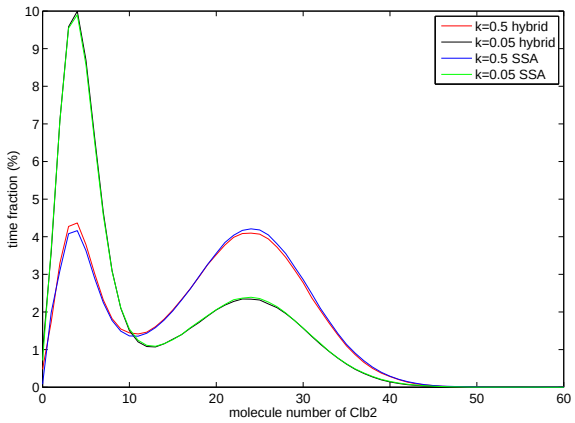


Figure 7.4: Distribution comparison between the hybrid method and SSA for bi-stable model. The y-axis is the time percentage that is occupied when $Clb2$ has a molecule number as the corresponding value on the x-axis. Each distribution curve is generated by one run of simulation for 5×10^6 time units.

As we can see that the hybrid method generates a much closer distribution to the exact numerical solution by the SSA when one compares Figure 7.3 with Figure 7.4. It is surprising to see the hybrid method match with the SSA so well even when $k = 0.05$ and the relaxation time is much higher than the average firing time from the slow subsystem. One would guess maybe it is because $Cdh1$ has a large population so that the stochastic effect of the chain reactions has little impact on $Clb2$. But that guess is not true. We reduced the total population of $Cdh1$ to 10 and even 5. In both cases the distribution generated from the hybrid method matches with the one from the SSA very well. We also reduced k even times smaller to 0.005. Their distributions were still close.

Chapter 8

Conclusions

In this thesis, our research is focused on solving stochastic simulation challenges in biochemical systems. The simulation challenge is presented in two scenarios: the multi-state and multi-scale situations.

In the first part, we have studied the multi-state feature of biochemical systems. We have analyzed the theoretical foundation of the StochSim and compared the SSA and the StochSim. We have demonstrated that when its step size is very small, the StochSim can be viewed as a first-order approximation of the SSA. In general the SSA is more efficient than the StochSim, especially when dealing with systems with a large total population. The StochSim has advantages when multistate species are involved in the system. Based on this analysis of comparison, we have proposed a hybrid method that possesses the advantages of both the StochSim and SSA. This hybrid method is tested with numerical examples and shows great

performance improvement. Besides the StochSim, rule-based modeling techniques have also shown advantages in representing systems involving multistate species [78]. Extending the Network-Free Algorithm, we have proposed a new hybrid method Population-NFA combining the population-based and particle-based schemes, and then proposed a new population-based method Full-Scale-SSA with sparse storage. Complexity analysis and numerical experiments on our new methods have shown that the PNFA and the FSSSA are accurate, but have efficiency differences depending on problem characteristics.

In the second part of the thesis, we have explored the multi-scale property of two stochastic eukaryotic cell cycle models. The first model is developed by Kar et al. In order to accelerate its stochastic simulation using Haseltine and Rawlings' hybrid method, we have proposed a more efficient partitioning strategy. Numerical experiments have demonstrated that, with our new partitioning strategy, the hybrid method accurately simulates intrinsic noise with improved simulation efficiency. A more serious difficulty with Kar et al.'s method is that it does not "scale up" easily to large deterministic models. We have proposed a new way of building stochastic models of the cell cycle. Assuming most of the intrinsic noise comes from the gene expression level, we build stochastic reactions at the gene expression level and preserve the original phenomenological ODEs from the deterministic model. This combined model is self-partitioned and easily simulated by the hybrid method. Following this idea, we have successfully built a stochastic cell cycle model based on the original three-variable model of Tyson and Novak [38].

The second model is the multiple site phosphorylation model which serves as the core compo-

nent of the cell cycle model developed by Barik et al. We have applied to this model Haseltine and Rawlings' hybrid method and the SQSSA which is an approximation method to deal with the multi-scale feature of a stochastic model. The SQSSA has shown its advantages in simplifying the target network by avoiding simulating the fast reactions and enhancing the overall simulation efficiency. However, its accuracy is only guaranteed when the supposed fast subsystem is really on a much higher time scale. To quantify this requirement, we have developed an efficient way to estimate the relaxation time for a chain reaction network, which is the typical subsystem for models based on multiple site phosphorylation. We have compared the approximated relaxation time of this particular chain reaction subsystem with the average firing interval of the slow reactions. Numerical experiments have shown that when the scale requirement is not satisfied, the SQSSA generates large errors. On the other hand, in this case the hybrid method, which simulates the fast subsystem with ODEs, has shown great accuracy.

Bibliography

[1] H. Miyata, M. Miyata, and M. Ito. The cell cycle in the fission yeast, Schizosaccharomyces pombe. *Cell Struct. Funct.*, 3:39–46, 1978.

[2] S. Di Talia, J. Skotheim, J. Bean, E. Siggia, and F. Cross. The effects of molecular noise and size control on variability in the budding yeast cell cycle. *Nature*, 448:947–51, 2007.

[3] J. Vohradsky. Neural model of the genetic network. *J. Biol. Chem.*, 276:36168–73, 2001.

[4] K. Chen, L. Calzone, A. Csikasz-Nagy, F. Cross, B. Novak, and J. Tyson. Integrative analysis of cell cycle control in budding yeast. *Mol. Biol. Cell*, 15:3841–62, 2004.

[5] N. Geard and J. Wiles. A gene network model for developing cell lineages. *Artif. Life*, 11:249–67, 2005.

[6] D. Chu, N.Zabet, and B. Mitavskiy. Models of transcription factor binding: sensitivity of activation functions to model assumptions. *J. Theor. Biol.*, 257:419–29, 2009.

[7] H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *PNAS*, 94:814–9, 1997.

[8] A. Arkin, J. Ross, and H. McAdams. Stochasitc kinetic analysis of developmental pathway bifurcation in phage λ-infected Escherichia coli cells. *Genetics*, 149:1633–48, 1998.

[9] N. Fedoroff and W. Fontana. Small number of big molecules. *Science*, 297:1129–31, 2002.

[10] M. Elowitz, A. Levine, E. Siggia, and P. Swain. Stochastic gene expression in a single cell. *Science*, 297:1183–6, 2002.

[11] S. Kar, W. Baumann, M. Paul, and J. Tyson. Exploring the roles of noise in the eukaryotic cell cycle. *PNAS*, 106:6471–6, 2009.

[12] D. Barik, W. Baumann, M. Paul, B. Novak, and J. Tyson. A model of yeast cell cycle regulation based on multisite phosphorylation. *Mol. Sys. Biol.*, 6:405, 2010.

[13] D. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.*, 22:403–34, 1976.

[14] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81:2340–61, 1977.

[15] C. Morton-Firth and D. Bray. Predicting temporal fluctuations in an intracellular signalling pathway. *J. Theor. Biol.*, 192:117–28, 1998.

[16] C. Morton-Firth, T. Shimizu, and D. Bray. A free-energy-based stochastic simulation of the Tar receptor complex. *J. Mole. Bio.*, 287:1059–74, 1999.

[17] T. Shimizu and D. Bray. *Computational cell biology - the stochastic approach in foundations of systems biology.* MIT Press, 2001.

[18] M. Sneddon, J. Faeder, and T. Emonet. Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nature Methods*, 8:177–83, 2011.

[19] C. Rao and A. Arkin. Stochastic chemical kinetics and the quasi-steady-state assumption: application to the Gillespie algorithm. *J. Chem. Phys.*, 118:4999–5010, 2003.

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

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

[22] E. Haseltine and J. Rawlings. Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics. *J. Chem. Phys.*, 117:6959–69, 2002.

[23] W. Hlavacek, J. Faeder, M. Blinov, A. Perelson, and B. Goldstein. The complexity of complexes in signal transduction. *Biotechnol. Bioeng.*, 84:783–94, 2003.

[24] D. Bray. Genomics: Enhanced: molecular prodigality. *Science*, 299:1189–90, 2003.

[25] H. Berg. *E. coli in motion.* New York: Springer, 2004.

[26] M. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A*, 104:1876–89, 2000.

[27] Y. Cao, H. Li, and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *J. Chem. Phys.*, 121:4059–67, 2004.

[28] T. Schulze. Kinetic Monte Carlo simulations with minimal searching. *Phys. Rev. E*, 65:036704, 2002.

[29] A. Chatterjee and D. Vlachos. An overview of spatial microscopic and accelerated kinetic Monte Carlo methods. *J. Comput. Aided Mater. Des.*, 14:253–308, 2007.

[30] A. Regev and E. Shapiro. Cellular abstractions: cells as computation. *Nature*, 419:343, 2002.

[31] A. Regev, E. Panina, W. Silverman, L. Cardelli, and E. Shapiro. BioAmbients: an abstraction for biological compartments. *Theoretical Computer Science*, 325:141–67, 2004.

[32] Z. Liu and Y. Cao. Detailed comparison between StochSim and SSA. *IET Sys. Bio.*, 2:334–41, 2008.

[33] Z. Liu, Y. Pu, and Y. Cao. A hybrid method combining improved StochSim with SSA for solving systems with multistate species. In *Proc. 2009 BIOCOMP*, 2009.

[34] W. Hlavacek, J. Faeder, M. Blinov, R. Posner, M. Hucka, and W. Fontana. Rules for modeling signal-transduction systems. *Science STKE*, 344:re6, 2006.

[35] L. Harris, J. Hogg, and J. Faeder. Compartmental rule-based modeling of biochemical systems. In *Proc. 2009 Winter Simulation Conf.*, 2009.

[36] Z. Liu and Y. Cao. Stochastic simulation methods for biochemical models with multi-state species. In *Proc. 2009 IEEE TIC-STH-SENCS*, 2009.

[37] Z. Liu, C. Shaffer, U. Mobassera, L. Watson, and Y. Cao. Multistate modeling and simulation for regulatory networks. In *Proc. 2010 Winter Simulation Conf.*, 2010.

[38] J. Tyson and B. Novak. Regulation of the eukaryotic cell cycle: molecular antagonism, hysteresis and irreversible transitions. *J. Theor. Biol.*, 210:249–63, 2001.

[39] Z. Liu, Y. Pu, F. Li, C. Shaffer, S. Hoops, J. Tyson, and Y. Cao. Hybrid modeling and simulation of stochastic effects on progression through the eukaryotic cell cycle. *J. Chem. Phys.*, 136:034105, 2012.

[40] Z. Liu and Y. Cao. Approximation of a multi-scale model based on multiple site phosphorylation. In *IEEE International Conference on Bioinformatics and Biomedicine Workshops*, pages 9–16, Philadelphia, PA, October 2012.

[41] J. McCollum, G. Peteron, C. Cox, M. Simpson, and N. Samatova. The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. *Comput. Biol. Chem.*, 30:39–49, 2006.

[42] N. Novere and T. Shimizu. Stochsim: modeling of stochastic biomolecular processes. *Bioinfomatics*, 17:575–6, 2001.

[43] M. Pettigrew and H. Resat. Modeling signal transduction networks: a comparison of two stochastic kinetic simulation algorithms. *J. Chem. Phys.*, 123:114707, 2005.

[44] V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable simulation of cellular signaling networks. *Lect. Notes Comput. Sci.*, 4807:139–57, 2007.

[45] J. Yang, M. Monine, J. Faeder, and W. Hlavacek. Kinetic Monte Carlo method for rule-based modeling of biochemical networks. *Phys. Rev.*, 78:031910, 2008.

[46] D. Gillespie. The chemical Langevin equation. *J. Chem. Phys.*, 113:297–306, 2000.

[47] D. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 115:1716–33, 2001.

[48] D. Gillespie and L. Petzold. Improved leap-size selection for accelerated stochastic simulation. *J. Chem. Phys.*, 119:8229–34, 2003.

[49] J. Ramshaw. Partial chemical equilibrium in fluid dynamics. *Phys. Fluid*, page 23, 1980.

[50] M. Rein. The partial-equilibrium approximation in reacting flows. *Phys. Fluid A*, 1992.

[51] L. Segel and M. Slemrod. The quasi-steady-state assumption: a case study in perturbation. *SIAM Review*, 1989.

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

[53] H. Salis and Y. Kaznessis. Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *J. Chem. Phys.*, 122:054103, 2005.

[54] J. Tyson. The coordination of cell growth and division: intentional or incidental? *BioEssays*, 2:72–7, 1985.

[55] M. Sabouri-Ghomi, A. Ciliberto, S. Kar, B. Novak, and J. Tyson. Antagonism and bistability in protein interaction networks. *J. Theor. Biol.*, 250:209–18, 2008.

[56] Z. Qu, J. Weiss, and W. MacLellan. Regulation of the mammalian cell cycle: a model of the G1-to-S transition. *Am. J. Physiol. Cell Physiol.*, 284:C349–64, 2003.

[57] D. Bray's Group. http://www.pdn.cam.ac.uk/groups/comp-cell/StochSim.html.

[58] L. Lok and R. Brent. Automatic generation of cellular reaction networks with Moleculizer 1.0. *Nat. Biotechnol.*, 23:131–6, 2005.

[59] O. Kapuy, D. Barik, M. Sananes, J. Tyson, and B. Novak. Bistability by multiple phosphorylation of regulatory protein. *Prog. Biophys. Mol. Biol.*, 100:47–56, 2009.

[60] J. Skotheim, S. Di Talia, E. Siggia, and F. Cross. Positive feedback of G1 cyclins ensures coherent cell cycle entry. *Nature*, 454:291–6, 2008.

[61] C. Hyver and H. Le Guyader. MPF and cyclin: modeling of the cell cycle minimum oscillator. *Biosystems*, 24:85–90, 1990.

[62] R. Norel and Z. Agur. A model for the adjustment of the mitotic clock by cyclin and MPF levels. *Science*, 251:1076–8, 1991.

[63] J. Tyson. Modeling the cell division cycle: cdc2 and cyclin interactions. *PNAS*, 88:7328–32, 1991.

[64] A. Goldbeter. A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. *PNAS*, 88:9107–11, 1991.

[65] M. Obeyesekere, J. Herbert, and S. Zimmerman. A model of the G1 phase of the cell cycle incorporating cyclin E/cdk2 complex and retinoblastoma protein. *Oncogene*, 11:1199–205, 1995.

[66] C. Thron. A model for a bistable biochemical trigger of mitosis. *Biophys. Chem.*, 57:239–51, 1996.

[67] M. Barberis, E. Klipp, M. Vanoni, and L. Alberghina. Cell size at S phase initiation: an emergent property of the G1/S network. *PloS. Comput. Biol.*, 3:e64, 2007.

[68] D. Zenklusen, D. Larson, and R. Singer. Single-RNA counting reveals alternative modes of gene expression in yeast. *Nat. Struct. Mol. Biol.*, 15:1263–71, 2008.

[69] J. Tyson and B. Novak. Temporal organization of the cell cycle. *Curr. Biol.*, 18:R759–68, 2008.

[70] M. Thattai and A. Oudenaarden. Intrinsic noise in gene regulatory networks. *PNAS*, 98:8614–9, 2001.

[71] P. Swain, M. Elowitz, and E. Siggia. Intrinsic and extrinsic contributions to stochasticity in gene expression. *PNAS*, 99:12795–800, 2002.

[72] J. Pedraza and J. Paulsson. Effects of molecular memory and bursting on fluctuations in gene expression. *Science*, 319:339–43, 2008.

[73] H. Salis, V. Sotiropoulos, and Y. Kaznessis. Multiscale hy3s: hybrid stochastic simulation for supercomputers. *BMC Bioinformatics*, 7:93, 2006.

[74] A. Hindmarsh. Odepack, a systematized collection of ode solvers, in scientific computing. *IMACS Transactions on Scientific Computation*, 1:55–64, 1983.

[75] D. Anderson. Incorporating postleap checks in tau-leaping. *J. Chem. Phys.*, 128:054103, 2008.

[76] B. Novak and J. Tyson. Design principles of biochemical oscillators. *Nature Reviews Mol. Cell Bio.*, 9:981–991, 2008.

[77] A. Berman and R Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Academic Press, 1994.

[78] J. Faeder, M. Blinov, and W. Hlavacek. Rule-based modeling of biochemical systems with bionetgen. In I. Maly, editor, *Methods in molecular biology: Systems biology*, volume 500, pages 113–167. Humana Press, 2008.