A Gillespie-Type Algorithm for Particle Based Stochastic Model on Lattice

Weigang Liu

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science & Application

Yang Cao, Chair
Shengfeng Cheng
Alexey V. Onufriev

Dec. 4, 2019
Blacksburg, Virginia

Keywords: Gillespie algorithm, stochastic simulation, lattice model

A Gillespie-Type Algorithm for Particle Based Stochastic Model on Lattice

Weigang Liu

(ABSTRACT)

In this thesis, I propose a general stochastic simulation algorithm for particle based lattice model using the concepts of Gillespie's stochastic simulation algorithm, which was originally designed for well-stirred systems. I describe the details about this method and analyze its complexity compared with the StochSim algorithm, another simulation algorithm originally proposed to simulate stochastic lattice model. I compare the performance of both algorithms with application to two different examples: the May-Leonard model and Ziff-Gulari-Barshad model. Comparison between the simulation results from both algorithms has validate our claim that our new proposed algorithm is comparable to the StochSim in simulation accuracy. I also compare the efficiency of both algorithms using the CPU cost of each code and conclude that the new algorithm is as efficient as the StochSim in most test cases, while performing even better for certain specific cases.

A Gillespie-Type Algorithm for Particle Based Stochastic Model on Lattice

Weigang Liu

(GENERAL AUDIENCE ABSTRACT)

Computer simulation has been developed for almost one century. Stochastic lattice model, which follows the physics concept of lattice, is defined as a kind of system in which individual entities live on grids and demonstrate certain random behaviors according to certain specific rules. It is mainly studied using computer simulations. The most widely used simulation method to for stochastic lattice systems is the StochSim algorithm, which just randomly pick an entity and then determine its behavior based on a set of specific random rules. Our goal is to develop new simulation methods so that it is more convenient to simulate and analyze stochastic lattice system. In this thesis I propose another type of simulation methods for the stochastic lattice model using totally different concepts and procedures. I developed a simulation package and applied it to two different examples using both methods, and then conducted a series of numerical experiment to compare their performance. I conclude that they are roughly equivalent and our new method performs better than the old one in certain special cases.

I dedicate this thesis to my beloved parents.

Acknowledgments

I am deeply grateful to my advisor, Prof. Yang Cao.

Contents

Chapter 1

Introduction

1.1 Computer Simulation

Since 1936 when Alan Turing proposed the principle for modern computer in his paper [1], it has been almost 100 years' development for this kind of computing devices. *Computer simulation* (or computer experiment), which is first time employed to a large-scale problem in the Mahattan Project in World War II, has also been developed rapidly side-by-side with the growth of modern computer. It approximately mimic the behavior of a system defined by a mathematical model using simulation codes. These models usually can not be solved either analytically or exactly. Therefore, computer simulation is widely accepted as another reasonable approach to study these models. The common feature shared by all types of computer simulation methods is that, they aim to generate a sample trajectory among all possible evolution history space, where a complete enumeration is impossible. One can estimate the reliability of the chosen mathematical model or study its properties using this method even if the analytic solution is too complex to obtain, as it can simply simulate events with scales that far exceed the calculation ability of traditional paper-and-pencil mathematical work. It

has been widely used in, for example, physics, astrophysics, climatology, chemistry, biology, ecology, psychology, health care and engineering, especially after the releasing of the fourth generation of computing hardware since 1971.

An important concept related to computer simulation is the *computer model*. It uses computational algorithms and mathematical equations to characterize behaviors of a corresponding system. Meanwhile, computer simulation are based on these equations and algorithms defined for a model. In another word, simulation is the process of running systems defined by models. There are various independent pairs of attributes possessed by computer models, such as steady or dynamic; stochastic or deterministic; continuous or discrete; local or distribute. According to these different attributes, computer simulation in science, for our own interests in this project, can be roughly classified as three different categories:

- Deterministic simulation: Numerically solve differential equations that do not have simple analytical solution. Models associated with this type of simulation are usually continuous and deterministic. This type of simulation can be applied in, e. g. theoretical physics, cosmology, fluid dynamics, continuum mechanics and chemical kinetics.

- Stochastic simulation, which is usually associate with discrete models that demonstrate certain randomness. Examples of related subjects are biology, biochemistry, condensed matter physics and ecology.

- Multiparticle simulation which, in contrast, deal with models of particles with continuous movement and demonstrating randomness, where the most important application is the molecular dynamics (MD) simulation.

There are also various other applications of computer simulation not listed above. Overall computer simulation can be viewed as one of the most important tools of research, after computer was invented, in a wide scale within the scientific community.

1.2 Monte Carlo method

Monte Carlo method represents a large group of simulation methods that use computer simulation with multiple stochastic simulations to study a system that demonstrates randomness. The modern Monte Carlo method was known to be first invented by Stanislaw Ulam in the late 1940s. John von Neumann then programmed the ENIAC computer to implement this method. It was first utilized during the nuclear weapon project in Los Alamos National Laboratory. Or to be more specific, it was used to investigate the radiation shield and the average distance that neutrons can travel through various materials. It was also the key point of simulations required for the Manhattan Project, relating to the earlier work of the hydrogen bomb's development. Later it became popular in many fields such as physics, mathematics, chemistry and engineering. Various of computational algorithms are known as the Monte Carlo method if they demonstrate the feature using a large set of random sampling to obtain numerical results. The essential idea here is to use randomness to solve problems that are hard or even impossible to approach in the deterministic case. However, these problems usually have a probabilistic interpretation as well, which, on another hand, is known to be a principle that Monte Carlo method is applicable. There are mainly three different classes of these problems: optimization, numerical integration and probability distribution generation. Many mathematics theory can be related to the Monte Carlo method, for example the law of large number, the ergodic theorem. The key point here is that, the sampled empirical measures are used to approximate the complex unknown distribution of the random variable. By the law of large numbers, these random empirical measures converge to the deterministic distribution of the object variable when the number of sampling approaches the infinity limit. In another word, with the Monte Carlo method, we can make reasonable approximation to these principle deterministic problems.

The definition about *Monte Carlo* has not been fixed. Many previous studies had tried to

at least distinguish some basic concepts about this term [2–4]. As pointed out by Kalos and Whitlock that even the computer code can be viewed as simultaneously as a 'natural simulation' or natural sampling, ambiguities may always exist. Anyhow, the development of the Monte Carlo method has been fairly rapid since the wide application of this method in many different fields.

Although there exists a broad class of variations, Monte Carlo methods usually follow a specific pattern:

- *Step* 1. Define the input domain and randomly generate inputs according to a probability distribution over this domain;
- *Step* 2. Perform specific computation on the inputs according to the predefined model;
- *Step* 3. Aggregate the results.

According to the law of large number, the accuracy of Monte Carlo simulation results will directly depend on the number of inputs we used here. In principle, more inputs are needed to give us better approximation. However, the quality of these random inputs generated from its distribution function is also an important point to judge our results. The list of "truly random" random numbers was first used when this method was just proposed. However, due to the fact that this strategy was extremely slow. von Neumann then developed pseudorandom number generator to improve the speed. Though this procedure was criticized as crude at the very beginning time. It is now widely accepted that the "truly random number" is not always necessary for Monte Carlo simulations. In fact, a deterministic, pseudorandom sequences is thought to be sufficient for most of the applications and simulations, as long as this pseudo-random sequence is "random" enough.

1.3 Stochastic lattice model

Although there are different definitions for the term *lattice* model, we are most interested in the one concerned in physics. In physics, a lattice model is a physics model defined on a lattice. An straightforward example is the crystal structure, where the regular distribution of atoms spontaneously suggest a lattice model. It is discrete and simple enough. It can also be viewed as an approximation to a continuum case, which is ideal in computational physics study. Some of the associated models are even theoretically solvable, a reason why it is also popular in theoretical physics.

The stochastic lattice model, which usually includes transition rules, is often used to characterize the nonequilibrium phenomena in physics, chemistry, biology or their cross subjects. These transition rules capture the microscopic interaction properties between the individual entities or agents. They can take different internal states and change their states according to the transition rules as time goes on. To study this kind of models, a most important method is the Monte Carlo simulation, as mentioned in preview section. To drive a system defined by stochastic lattice model, there are exist many different policies. Due to the particle-based or agent-based essence of lattice model, most of the existing simulation algorithms are also particle-based, for example, the StochSim algorithm [5]. These particle-based or object-oriented method are simple programming and thought to capture enough details about the associated system. However, a naturally raised question here is that, if other alternatives that can also simulate this kind of model exist? Furthermore, if such a method is found, will it provide us identical or different results compared with that obtained by the traditional particle-based method? What the advantages and disadvantages of this alternative? We want to answer these question through this thesis.

1.4 Structure of this thesis

This thesis is structured as follows. In Chapter 1, we provide a general introduction about the research topic in this thesis and propose the question we want to fix. A deeper review of the studies associated with this topic is given in Chapter 2. We answer the first part of our questions in Chapter 3 by generating an "reaction-based" Gillespie-type algorithm that can be used to simulation stochastic lattice model. We clearly describe the general simulating procedure of this method and theoretically discuss the efficiency of this method when compared with the widely employed StochSim algorithm for stochastic lattice model. We then select two example models and implement them using both the original StochSim algorithm as well as our new proposed Gillespie-type algorithm in Chapter 4. We carefully compare their simulating results and efficiency to answer the second part of the our question. Finally, we make our overall conclusion and overlook about this research topic in Chapter 5.

Chapter 2

Background

2.1 Introduction

The history of describe biochemical systems using reaction rate equations can be traced back to nearly two centuries ago. Math in this description is continuous and deterministic, which is controversial and not universally accepted, since particles are discrete and randomness should be important in describing their behavior. Nearly two decades ago, discreteness and stochastic effect are shown to be important in microscopic systems consists of small numbers of reacting molecules, such as living cell [6–9]. More and more attention are paid to modeling the intrinsically discrete stochastic behavior, especially the simulation method using computer.

In fact since 1970s, Gillespie had devised an exact method to stochastically simulate chemically reacting systems on computer. Molecule species inside those systems are thought to be well-mixed, thus no spatial features emerge, and interact with each other follow a specific set of reaction rules. Their time evolution can be solved within the frame predicted by the stochastic chemical master equation and take the form of a Markovian random walk in the

N-dimensional space expanded by the population of each reacting species. Here, N is the number of different kind of species. This simulation method is known as Gillespie's stochastic simulation algorithm (SSA). There are various progresses have been made among those four decades after this algorithm was proposed to improve the accuracy or efficiency, for example the approximation version:tau-leaping method[10]. Applications and extensions of the SSA are also widely studied, such as to investigate not well-stirred system, there exists an extension of the standard SSA to reaction-diffusion processes in spatially inhomogeneous systems[11].

However, there is another simulation method: the stochastic simulator (StochSim), which is proposed by Morton-Firth and Bray [5]. Different from SSA, it has object-oriented nature. Therefore, it is thought to be more extendable for dealing with spatially inhomogeneous case. We will then review the studies considering algorithms to simulate those discrete-stochastic chemical systems, which are important for understanding the problem studied by us.

2.2 SSA for well-mixed system

Since 1967 when McQuarrie first gave an overview of what is now known as *chemical master equation*(CME)[12], the mathematical basis for modeling the intrinsically discrete-stochastic behavior of chemical systems was developing and widely known. In 1976, a stochastic simulation approach to biochemical reactions was developed by Gillespie [13, 14]. It is known as stochastic simulation algorithm (SSA) and oriented to well-stirred dilute chemical systems. The microphysical promise here underlies the CME[15], which is thought to be a more realistic description of chemical reaction systems than the original continuous-deterministic first order ordinary differential equation description. Two types of reaction are primarily

considered in this algorithm, either unimolecular or bimolecular, which contain one or two reactants and generate something else. Trimolecular and other types of reaction can be approximate by a series of two or more uni- or bi- molecular reactions.

Details about SSA can be introduced as follow. Suppose the system has N reaction species $\{S_1, S_2, ...S_N\}$ and their molecules are included in M reaction channels $\{R_1, R_2...R_M\}$. We can then use $\boldsymbol{X}(t) = (X_1(t), X_2(t)...X_N(t))$ denotes the state vector, where $X_i(t)$ is the number of molecule S_i at time t, and $\boldsymbol{v_j} = (v_{1j}, ..., v_{Nj})$ denote the state change vector, where v_{ij} is the change in of the number of species S_i caused by one single R_j reaction event. v_{ij} is supposed to be signed integer or zero. Propensity function a_j is also needed to fully characterize a reaction R_j. It is defined to be such that, $a_j(\boldsymbol{x})dt$ gives the probability that a R_j reaction will happen during the next time interval $[t, t + dt)$ from time t, if given $\boldsymbol{X}(t) = \boldsymbol{x}$. The detail formula of the propensity function for those two reaction types considered by this algorithm is also shown in Gillespie's studies [13, 14]. For a unimolecular reactions R_j with single reactant S_1, the probability that a randomly chosen reactant molecule will react in next dt is always proportional to dt with a constant coefficient c_j. If we sum $c_j dt$ over all x_1 S_1 molecules, in according to the superposition property of the probability, we have $a_j(\boldsymbol{x}) = c_j x_1$. The propensity funcion for a bimolecular reaction R_j is also given according to a simple kinetic theory argument [13, 14]. If those two reactants of R_j, S_1 and S_2 , are of different species, the propensity function is $a_j(x_1, x_2) = c_j \cdot x_1 x_2$, with x_1, x_2 are number of molecules of species S_1, S_2. Otherwise, in the two reactants of same species S_1 case, $a_j(x_1) = c_j x_1(x_1 - 1)/2$. The propensity function for a "spontaneous creation" or "external source" reaction is also considered, which is merely a constant reaction parameter $a_j(\boldsymbol{x}) = c_j$.

With all those formulation above, we can then introduce the computational procedure of SSA. The kernel of this simulation method is to give the joint probability density function (PDF), $p(\tau, j|\boldsymbol{x}, t)$ such that $p(\tau, j|\boldsymbol{x}, t)d\tau$ gives the probability, when $\boldsymbol{X}(t) = \boldsymbol{x}$, that the

next reaction event will happen during $[t+\tau, t+\tau+d\tau)$ and will be an R_j. The detail formula of this PDF is shown in Gillespie's study[13]:

$$p(\tau, j|\boldsymbol{x}, t) = e^{a_0(\boldsymbol{x})\tau} a_j(\boldsymbol{x}) \tag{2.1}$$

where $a_0(\boldsymbol{x}) = \sum_{k=1}^{M} a_k(\boldsymbol{x})$. The simulation procedure can be listed as follow:

- *Step* 1, Calculate $a_1(\boldsymbol{x}), ..., a_M(\boldsymbol{x})$ and their sum $a_0(\boldsymbol{x})$
- *Step* 2, Randomly determine τ and j according to the PDF (2.1)
- *Step* 3, Up date the state $(\boldsymbol{x}, t)$ by $(\boldsymbol{x} + \boldsymbol{v_j}, t + \tau)$
- *Step* 4, Return to step 1, or end the simulation

There are various method to implement step 2 here. The earliest two are proposed by Gillespie's paper [13], which is known as *direct method*, which is according to a "conditioning" decomposition of the PDF (2.1) and the *first reaction method*, which is shown to be equally exact as the direct method. However, due to the fact that first reaction method will discard $M-1$ unused random number each individual step, its efficient is thought to be much less than the direct method.

There also exist other implementation of step 2 with different method subsequently developed by other workers. For example. Gibson and Bruck[16] successfully extend the first reaction method into an equivalent but more efficient one which is called *next reaction method*. It is believed to be more efficient than the direct method, especially for systems with many reacting species and loosely coupled reaction channels. Cao et al. [17] had shown that for a very specialized class of problems, the direct method will be most efficient one among all those three. They even proposed the *optimized direct method*, which perform even better than the original direct method. Other than those, there are the *first family method* generated

by Lok[18]; the *sorting direct method* generated by McCollum et al. [19]; the *modified next reaction method* generated by Anderson[20]; and the *composition-rejection method* proposed by Slepy et al. [21]. Gillespie et al. [22] also pointed out that SSA can handle delayed event much easier than that using CME.

Other than those efficiency improvement methods, there are also some previous studies tried to deal with special cases. For example, the *slow-scale stochastic simulation algorithm*(ssSSA) is proposed by Cao et al. [23, 24] to solve the inefficiency exists in the stochastic stiffness problem. the *weighted* SSA, introduced by Kuwahara and Mura [25]; the *unweighted* SSA, studied by Gillespie et al [26]; and the *state-dependent doubly weighted* SSA, invented by Roh et al [27] are aiming to solve the difficulty that standard SSA is ill-suited to quantifying rare event problems. Rao and Arkin also discussed to apply the quasi-steady-state assumption to the SSA and gave a modified Gillespie algorithm [28].

2.3 Tau-leaping for well-mixed system

To improve the efficiency of SSA while sacrifice acceptable accuracy, in 2001 Gillespie introduced a faster but approximate stochasitic simulation procedure called the *explicit Poisson tau-leaping method* [10]. The basic idea of this method is to "leap" the system by a *preselected* time τ (not the same as the τ in original SSA) which is large enough to encompass more than one reaction events, as well as small enough that

$$a_j(\boldsymbol{x}) \approx \textit{constant in } [t, t+\tau),\ \forall j \tag{2.2}$$

This is the 1st leap condition. As along as this restriction is satisfied, A Poisson random variable with mean $a_j(\boldsymbol{x})\tau$ for each reaction R_j is generated to give the number of reaction

R_j events that will occur during $[t, t+\tau)$. Therefore, we update $\boldsymbol{X}(t)$ by:

$$\boldsymbol{X}(t+\tau) \doteq \boldsymbol{x} + \sum_{j=1}^{M} \mathcal{P}_j(a_j(\boldsymbol{x})\tau)\boldsymbol{v}_j \tag{2.3}$$

given $\boldsymbol{X}(t) = \boldsymbol{x}$ and the $\mathcal{P}_j$ are M independent Poisson random variables. However, there are still some problems one may encounter when apply this method. First of all, it is possible that one select a leap time that is too small that the M Poisson random variables in (2.3) are all zero. System state will then stay unchange and this method will be infinitely inefficient in this case. The police for selecting the leap time τ is discussed by Gillespie and Petzold [29], as well as by Cao et al. [30]. Another problem of the explicit tau-leaping procedure is that, since Poisson random variable can have arbitrary large sample value, there is always possible that one or more reaction channel fires too many times that some reactant species population will be driven negative. Modification of the original algorithm to avoid this unrealistic situation is provided by Cao et al. [31]. There are other variations of the tau-leaping method to deal with special problems or improve the efficiency. For instance, the *D-leaping* method is proposed by Bayati et al. to simulate biochemical systems with delays[32]; the *R-leaping* and *ER-leaping* method are designed by Auger et al. [33] and Mjolsness et al. [34] to speedup over SSA through either pre-selecting total number of reaction firing instead of the time, or introducing rejection sampling concept; the *unbiased post-leap rejection* procedure, proposed by Anderson [35] to rule out the bias as well as the negative population problem; and the *implicit tau-leaping* method of Rathinam et al. [36–39].

2.4 StochSim algorithm for well-mixed system

Another important stochastic simulation method is the object-oriented stochastic simulator(StochSim). It is first developed by Morton-Firth and Bray to simulate individual molecules in the cell signalling pathway[5, 40]. It also reduces all possible reaction types to uni- and bi- molecular reactions[41, 42], just like SSA did. However, instead of reaction rate, it directly employs reaction probability to determine whether a reaction can successfully occur or not. All the reaction probabilities are precomputed based on user defined rate constants. For those situations considered by us in this project, only the ratios between different rates are really important. Different rate set with same ratio can always match with each other through rescaling the time variable through some constants. A look-up table is used to store these reaction probabilities of all reaction channels. The rows and columns represent the first and second reactants, respectively. Besides real molecules, there are pseudo-molecules listed as the second reactant, which ensure a uniform simulation procedure for two types of reaction considered here. An entry storing a non-zero probability in this table means a reaction channel between those two reactants. If there multiple reaction channels between same reactants, the sum of the probabilities of all these channels are stored. Additional step is needed to deal with this degeneracy condition, which will be discussed in the simulation procedure.

StochSim proceeds discretely with equally divided time steps for each Monte-Carlo step (MCS). The simulation procedure of a individual step is described as follow:

- *Step* 1, Uniformly and randomly select the first reactant from through all real molecules

- *Step* 2, Uniformly and randomly select the first reactant from through all real and pseudo molecules except the one selected in *Step* 1

- *Step* 3, Search the look-up table to find the corresponding reaction probability. If a non-zero probability is found, a uniform random number in $(0.0, 1.0)$ is generated to compared with the non-zero probability. If it is smaller than the associated probability, there is a reaction between these two species. Otherwise, including the situation that the founded probability is zero, no reaction occurs in this individual step. If in the degenerate case, another random search according to the probability ratios between all possible reaction is needed again to exactly determine which reaction will occur.

- *Step* 4, Update the system according to the reaction rule and decide if proceed to next simulation step or end up simulation.

The number of individual steps in each MCS is equal the number of real modelcules at the beginning of this MCS, limited by the example model considered by this thesis. However, more philosophical and chemical-oriented discussion about the time step selection as well as the reaction computing had been reviewed by Chaterjee and Valchos [43].

Various previous studies have discussed the difference and consistency between the simulation algorithms StochSim and SSA. It was originally claimed by Shimizu and Bray [44] that these two algorithms are based on equivalent physical assumptions. Later on, Pettigrew and Resat [45], on another hand, had shown the efficiency difference between these two simulation method. Liu and Cao [46] then presented a comparison of these two algorithm from both accuracy and efficiency aspects and showed that the StochSim can be viewed as a first-order approximation to the corresponding SSA if the time step used in the StochSim is small enough. A hybrid simulation algorithm is also proposed by them to combine the advantages of both algorithms.

2.5 SSA beyond well-mixed system

The well-mixed assumption is not satisfied for many situations, particularly when inhomogeneity is introduced. Thus, spatial resolution as well as stochasticity are both required in the simulation. The SSA need to be modified. Trace back to 1970s, an extension of CME to the spatial inhomogeneous case is the reaction-diffusion master equation (RDME), which was proposed by Gardiner et al. [47]. It subdivides a system volume into multiple subvolumes in such a way that the molecules within each subvolume can be viewed as well-mixed and only react with other molecules inside the same subvolume. The diffusion between adjacent subvolumes is modeled as a "diffusive transfer reaction", which simplifies the original RDME problem to a well-mixed CME through reinterpreting the symbols used to describe the system. It is called mesoscopic reaction-diffusion systems. Details about this method are studied by Stundzia and Lumsden [11] and will be further discussed in the next chapter. One important issue that should be clarified before starting simulation is about how to divide the subvolumes here. Uniform Cartesian meshes are efficient and straightforward for systems with simple geometries. However, it usually fails in cellular or subcelluar structure cases, which have curved inner and outer boundaries. Various studies have considered different discretization strategies [48–51] as well as how to lower the complexity in selecting the next event [52] as the number of reactions increase vastly if there are many subvolumes.

Chapter 3

Algorithms Description and Comparison

3.1 Introduction

As described in the previous chapter, we are interested in the simulation of stochastic nonlinear chemical reaction models on spatial lattices, where diffusion has to be included in addition to the well-stirred assumption, on which the traditional Gillespie algorithm is based on. As we have introduced, there exist two important methods of stochastic modeling on the evolution process of biochemical homogeneous systems, namely Morton-Firth and Bray's stochastic simulator (StochSim) [5, 40] and Gillespie's stochastic simulation algorithm (SSA) [13], where Monte Carlo techniques are adopted to simulate the Markov process described by the associated master equation. However, since spatial dimensions are included in our model, we have crossed into the regime of stochastic reaction-diffusion systems and there will be special features involved with diffusion. To appropriately simulate this kind of systems, we need simulation methods different from their well-stirred peers. A straight for-

ward thought here is that we can extend the original simulation methods, namely StochSim and SSA, to include the spatial diffusion events in our model. Various studies have been done based on this idea and have shown consistency with theoretical predictions. However, there are still certain limits that exist on those generalized algorithms, which may put a restriction on systems to be simulated. We will first generally describe the lattice model we consider in this study and briefly further review some of them in the next section for both their advantages and disadvantages and based on those facts, propose our own method, which may break some of the limits original algorithms bearing. We will describe both the prototype and an improved version of our algorithm in detail, which is generated by following mainly Gillespie's simulation concepts. We will also present a comparison between this method and the corresponding StochSim version algorithm, which is widely employed in relevant studies.

3.2 General problem formulation

First of all, we should make definitions mentioned above more clear. By the name "lattice model", we refer to it in the context of condensed matter physics and/or computational physics. It is a physics model defined on a lattice (for example, triangle, square or hexagonal). It is a discrete model and can be thought of as a simplified network. Particles can occupy different sites on it, diffuse along edges connecting its neighbor sites or react with other particles (or just react itself) that are *available*.

We will describe more details about which reaction can be considered to be "available" as we mentioned above. There will be two factors determine whether it is available or not. First, one can be associated with our formulation of the general problem we want to address here. Let us suppose that we are working on a lattice with N sites which can contain active particles of M species $S_j(j = 1, ...M)$. We can also include inert species here, for example,

the empty site case may be considered as a passive particle in some models. Thus the state of a lattice can be represented as $\{X_1, X_2...X_N\}$, $X_i = x_{i1}, x_{i2}...x_{iM}$, x_{ij} is then thought to be the current number of particles of species S_j on-site i. Furthermore, we assume that those M species can participate in K different reactions $R_u(u = 1, ...K)$, store them in a look-up table for all those possible reactions, each can be one of the following general cases:

$$\begin{aligned} \varnothing &\xrightarrow{c_e} S_j \\ S_j &\xrightarrow{c_s} S_k(or\ \varnothing) \\ S_j + S_k &\xrightarrow{c_d} S_l(or\ \varnothing) + S_m(or\ \varnothing) \end{aligned} \tag{3.1}$$

Here $\varnothing$ means nothing or inert particle empty sites and the first reaction above can be called "external source" reaction and c_e, c_s, c_d can be understood as either reaction rates or simply the possibility whether one selected reaction can happen or not (will be explained in later subsection). There can also be reactions with three or more reactants situations. But for our own consideration in this study, we won't worry about them as the original StochSim algorithm for well-mixed homogeneous systems are free from these issues. Therefore, our systems will be driven by those reaction rules (diffusion processes are also included as $S_j + \varnothing \xrightarrow{c_d} \varnothing + S_j$, which means just switch the sites they were occupying). In other words, a reaction is available or not should first be determined by whether the two corresponding particles (or single-particle) are reactants (reactant) in those K reactions we assumed above. Secondly, it will depend on our model formulation, specifically for two-reactant reactions. There are also existing different assumptions about when two particles are available for reaction. Two of them are most popular when studying those lattice models. One is the on-site reaction assumption where one assumes that only when two particles are occupying the same site, they can react. Another one is the nearest-neighbor reaction case that when two particles are nearest neighbors they can react. The latter assumption is often made

together with site occupation restriction, especially when one only allows at most one active particle in one site. This single-occupation restriction will also rule out the only on-site reaction possible case. Furthermore, they both assume that the reaction products will still occupy the same sites as the reactants. Details about how to arrange them will be defined by the specific reaction. For example, if occupation restriction is introduced, we really need to consider the number of products about a specific reaction thus if there is enough space for them or not. However, each of these two assumptions may be reasonable for specified limited reaction-diffusion processes and may even be equivalent then. Here for our own research interests and to make things easier to understand, we will restrict our scope in the only nearest-neighbor reaction allowed the case with at most single-particle occupation assumption. The product number problem is spontaneously ruled out under this assumption if one further views the empty site as an inert particle and keep the total number of all kinds of particles conserved, which again is our consideration here. We also assume that we are working with a two-dimensional square lattice case, which can be easily extended to a higher dimension and other kinds of the lattice. We will not talk about how to initialize the system in detail, as one can either assume some specific initial configuration or even prepare it totally randomly. We will only focus on the description of how do we simulate the time evolution of the lattice state, knowing the initial configuration already. That's our general formulation.

3.3 Motivation

A crucial difference between the well-stirred homogeneous systems and spatial lattice models is that each particle is then distinguishable with each other even if they are in the same species in the lattice model case, in contrast to the well-stirred case. In other words, the

state vectors used in the SSA method will have a dimension equal to the number of different spatial sites that assumed in our lattice model rather than the number of different species in the original case. Therefore, the object-oriented algorithm is thought to be better performed when one faced with this specific kind of system. Furthermore, the stochastic feature is intrinsic here within or between species, which will need to be introduced additionally in SSA among particles of the same species. There exists a generalized StochSim algorithm considering this stochastic lattice model, which will be discussed in detail.

3.3.1 Generalized StochSim algorithm for stochastic lattice model

With all those assumptions we made above in Sec. 3.2, our detailed steps in simulating an initialized stochastic lattice system can be outlined as follows:

- *Step* 0(initialization). Specify an initial configuration for a lattice system and the reaction parameters $c_1, c_2, ...c_M$ series for reaction$\{R_u\}$, store them, determine the time increment or follow the policy described in *Step.* 3 and then set the time variable $t = 0$.

- *Step* 1. Randomly select a lattice site, if it is occupied by an inert particle, re-select again until one active particle is found and then uniformly and randomly select one nearest neighbor of it (the second select reactant or the neighbor particle can be inert, in another formulation one may select an inert particle which won't make things different indeed) or select none of them (which means single reactant reaction).

- *Step* 2. Search in the reaction look-up table for the possible reaction between those two select particles. If none exists, one directly finishes this time step and increase the time variable with a specific time increment. Otherwise, if a reaction R_u is found, one needs another uniform random number among $(0.0, 1.0)$ and compared it with

the corresponding reaction probability (rate) parameter c_u to determine whether this reaction can happen. If so the next step is to change the lattice configuration according to the rule, one assumes in the specific formulation and then increases the time varies according to specific time increment. It is possible that multiple reaction channels exist between the same reactants (or reactant), one may need one more random number to determine which reaction is to be checked.

- *Step* 3. Check if one needs to recalculate the time increment or not. If so, regenerate it; otherwise, return to *Step*. 1. A general procedure used in relevant studies is that one Monte Carlo Step (MCS) is considered completed when on average all active particles have been selected once. Thus, roughly, one can first calculate how many iterations are needed for $\Delta t = 1.0$ according to current lattice configuration, run all of them, re-determine how many iterations for the next 1.0 time interval again and then iterate until the preset time wall is reached.

To illustrate this algorithm more clearly, we employ a simple model with one active species A and three relevant reactions. In our previous assumption, as single-particle occupation restriction is also assumed here, the empty site can be also viewed as an inert particle in this formulation. Those three reactions are listed as follows:

$$\begin{aligned} A &\xrightarrow{a_1} \varnothing \\ A + \varnothing &\xrightarrow{a_2} A + A \\ A + \varnothing &\xrightarrow{a_3} \varnothing + A \end{aligned} \tag{3.2}$$

The third reaction above here means A hopping to the empty site and left it originally

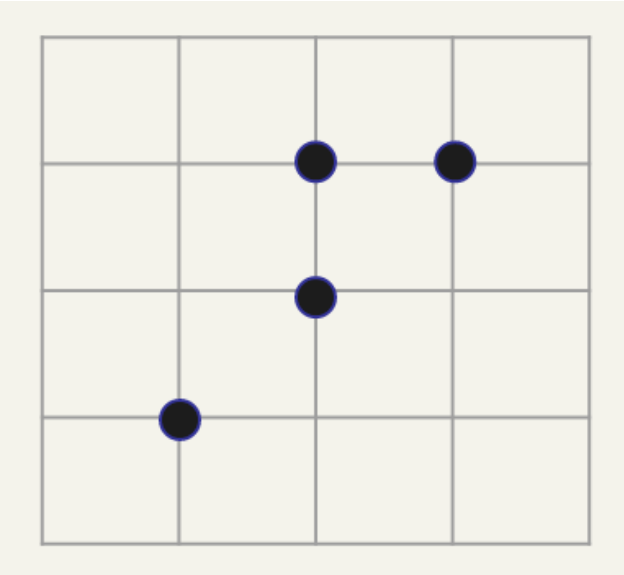

Figure 3.1: An example 5×5 square lattice with 4 sites are occupied by particle A (black dots)

occupied site empty. An example 5×5 lattice configuration is also given in Fig. 3.1, which can be thought of as an initial configuration.

To investigate the evolution of this system, we first set $t = 0.0$ and follow the time incremental procedure described in *Step.* 3 above, for a time interval between $t = 0.0$ and $t = 1.0$, we have 4 active particles here, thus we will try to loop *Step.* 1 and *Step.* 2 four times. Suppose that we first select the upper left corner of this lattice, which is not occupied (thus it is an empty site or inert particle), we need to re-select until we find an active particle here (one of the black dots). Let us assume we finally find the most central lattice site, which is occupied by a particle A. We then randomly select one of its four neighbors or none of them (with probability 20% for each case). If we choose upper, there is also an active particle A in that site. However, if we go back to the reaction look-up table 3.2, there exists no reaction between two A, thus this loop directly finished. Or if we select another three neighbors, they all give us a $A + \varnothing$ pair. Go back to the reaction table, there are two reactions with exactly the same reactants as what we have here. Thus we should decide which one can happen (with equal

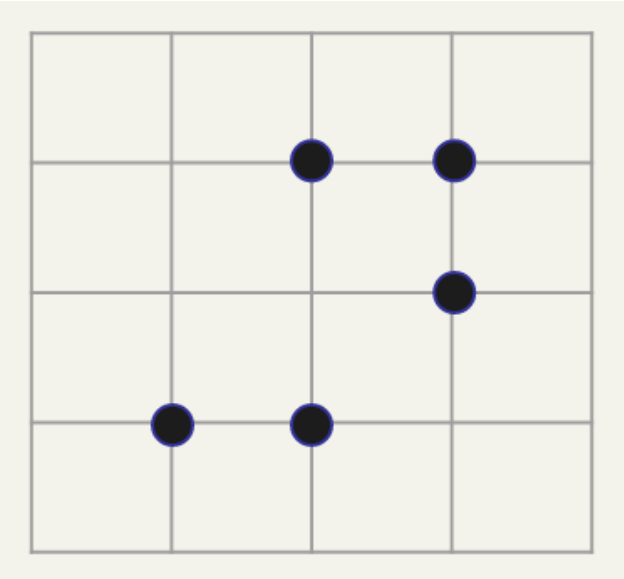

Figure 3.2: An example 5×5 square lattice with 5 sites are occupied by particle A (black dots)

probability) and then decide if it can happen. If the reproducing reaction is selected (the second one), and our uniform random number is smaller than our preset reaction probability a_2, reaction then happen and we should fill the selected empty site by a new particle A. Otherwise, we select none of its four neighbors, this selected particle A is the only reactant and we check the look-up table find the first reaction include only one A as its reactant. We then follow the same procedure as a two-reactant reaction case, if the reaction does not fire, nothing happens and move to the next loop; or if it happens, we just remove the particle A in the most central site and leave with an empty site. Let us then assume that after 4 loops finished, we update our time variable by $t = 0.0 + 1.0 = 1.0$, our lattice configuration then becomes like Fig. 3.2. We have 5 active particles then, thus we need to try 5 times in the time interval between $t = 1.0$ and $t = 2.0$ is five.

To this end, we have introduced the generalized StochSim algorithm for simulating a stochastic lattice model with a single-occupation restriction. It has both advantages and disadvantages, which we will briefly talk about.

On one side, due to the fact that the original StochSim is invented to be an object-oriented algorithm, the implementation is easy enough and is supposed to maintain high efficiency compared with those generalized SSA methods which will be talking about in the next subsection, either studied by another group or proposed by us. Although in our formulation, we assumed the only nearest-neighbor reaction case, either of those assumptions and even their hybrid formulation can be implemented in through this algorithm, again in contrast to the SSA cases which will be discussed later.

However, its disadvantage can be viewed mainly from two aspects. First of all, as in *Step.* 1 above, one needs to randomly find an active particle first, which may cost much when the system is low occupied. This can be cured through establishing an additional data structure that stores the coordinates of all the active particles current exist in the system. One can then directly search in this mapping structure rather than randomly search in the lattice with reject policy, as described in *Step.* 1 above. An additional cost is needed to maintain this mapping structure to be consistent with our lattice configuration. Other than that, it is shown by a previous study [46], the original StochSim for well-stirred systems can be viewed as a first-order approximation to the SSA, which is thought to be an exact simulation method that is consistent with chemical master equations in resulting distribution. Therefore, in the well-mixed homogeneous case, StochSim may perform differently compared with SSA, especially when the population between different species can vary over different order or the reaction rates (probabilities) are of a different order (for instance in reaction lookup table 3.2 above, if $a_1/a_2 \approx 10.0$). Apparently, as for the generalized StochSim is thought to bear the same problem as we just extend the original model to include the spatial feature, which actually can be understood as we add our spatial lattice as an additional restriction in original searching step to decide if there exists the corresponding reaction.

From these considerations, a generalized SSA algorithm that can be employed in the stochastic lattice model is necessary, especially when one needs to handle reaction rates or popu-

lations of multiple orders. We will briefly introduce previous consideration in the next subsection, as well as point out their limitation.

3.3.2 Mesoscopic reaction-diffusion systems simulated using SSA

The previous treatment of the simulation of non-linear reaction-diffusion processes is to introduce the mesoscopic diffusion rates that can be formally analogous to reaction rates. Through appropriate mapping from the diffusion coefficient to the transition rate probability for the diffusion of the individual particle between finite subvolume, one can then subdivide the original into some subvolumes or 'voxels' and then consider a model of diffusion whereby each subvolume exchanges particle with its nearest neighbors only through reaction-like processes. In other words, the diffusion process is thought to be an additional reaction that happens between different subvolumes. For example, if we go back to the simplest model we proposed in the previous subsection, where we have a species A, we first subdivide our whole system into N different subvolumes in one dimension (thus those N subvolumes are connected with each other like a chain). We can then rewrite species A as $\{A_1, A_2, ...A_N\}$, A_i means species A in subvolume i. Then including the diffusion feature just means that we need to add a set of reactions into our reaction table:

$$A_i \overset{k_{i,i+1}}{\rightarrow} A_{i+1}, A_i \overset{k_{i,i-1}}{\rightarrow} A_{i-1} \tag{3.3}$$

Furthermore, one will also assume that our original reaction set 3.2 can only happen within one subvolume, which can be thought of as the lattice model in the last subsection with only on-site reaction assumption. The subvolume here can be understood as a lattice site without an occupation limit. To be more specific, here we divide our original species A into

N different species $\{A_i\}$ and thus we have $3 \times N$ different reactions that come from our original model, as well as $2 \times N$(if the periodic boundary condition is assumed) reactions that resulted from the diffusion processes analogy. After finish those work, this mesoscopic reaction-diffusion systems simulating method using the Gillispie algorithm can be exactly the same as the original SSA but with a much larger number of reactions than the original well-stirred systems. It is said that the number of reactions can easily be several million even a modest mesh to subdivide the systems with a small set of reactions. We will not go through much more details about this method since it is established based on a totally different assumption from our interests. The advantage of this method is that it can be easily developed for chemical kinetics simulation studies that incorporating diffusion feature for even complex geometries or considering inhomogeneous reaction-diffusion processes. However, its disadvantage or limitation of this method is clear. First of all, as claimed above, the number of reactions can vastly increase as one want to increase the mesh size. Secondly, it is pointed out by Stundzia et al, this algorithm is applicable for systems with collisions between inert and reacting species occur more frequently than collisions between reacting species, which may be not our case, as we can clearly see in Section 5, where we present some examples about the systems that we are interested.

To this end, as we have introduced those two previously studies generalized algorithms, we can then conclude our motivations here. Both of those simulating methods are reasonable generalizations of the original StochSim and SSA. Each of them has its own applicable regime and is proven to perform well there. However, from the disadvantages of each method we had listed above, we can easily find that there is a vacuum regime which is not covered by either of those two existing generalization algorithms, namely a system of approximately similar chance of collision between all the particles while the reaction rates (either of real reactions or diffusion analogy reactions) or particle populations are of different orders. We then aim to develop an algorithm that can deal with this case. We will discuss our own

method in the next subsection for both the prototype and our optimization versions, which can be understood as another generalization of SSA.

3.4 Generalized SSA algorithm for stochastic lattice model

Let us first review our simplified question here, we want to simulate a stochastic reaction-diffusion system on the 2D square lattice, we have at most single-occupation restriction for lattice sites and thus only nearest-neighbor reaction assumption. By viewing the empty site as an inert particle, we also suppose total particle number conservation here. Furthermore, to anchor our problem, we assume that the reaction rates (probabilities) or particle populations vary between different orders where the generalized StochSim for lattice model may perform different from the master equations suggestion. The later one suggests us to use SSA as a start point. We will then describe our algorithm that can tentatively solve our question then.

3.4.1 Algorithm prototype

As the question we are faced here can be solved by the generalized StochSim algorithm which we have talked about in the previous section as long as the multiple order reaction rates or particle populations restriction is removed. Thus a straight forward thought here is that, if we can directly substitute those operations that follow the well-mixed StochSim concepts with SSA, then we are done. To fulfill this assignment, we need to carefully compare the original StochSim and SSA. As we have pointed out in the previous section, an important difference between SSA and StochSim is that: StochSim is an object-oriented algorithm. Thus, it can

be generalized to solve the stochastic lattice model problem easily as this model is thought to be particle-based, or in other words, object-oriented as well. On the other hand, SSA can be thought of as a "reaction-based" algorithm. To apply it in a particle-based model, we first need to think about how to transform between these two pictures, specifically for our lattice model here, as the most important issue for SSA is how to correctly generate the propensity function.

Let us first consider the original SSA, in the well-stirred case, we generate the propensity functions are generated through those formulas:

$$\begin{aligned} a_e &= c_e, \\ a_s &= c_s * x_j, \\ a_d &= c_d * x_j * x_k \end{aligned} \tag{3.4}$$

From top-down, they are propensity functions for "external source", single-reactant and two-reactants reactions. We have stopped here since in the lattice model we consider here higher-order reaction can be ignored. The first two formulas can be directly applied in our lattice model as there is no spatial information embedded. However, we need to consider the third one, namely the two reactant reactions, carefully in our case as in a lattice model with only nearest-neighbor reaction assumption, we really put restrictions on it. There are three multipliers on its right-hand-side: the first one is the preset reaction rates, which would not change much in our case; the second and third together give us the probability that one pair of those two reactant meet each other in this well-mixed homogeneous system if we further divide it by a factor of V^2, where V is the volume of our systems. It is not the case for our lattice model with only nearest-neighbor reaction-restriction, where we assume that only when two reactants are the nearest pair, they can react with each other. In other words, only two particles share one lattice edge are thought to be possible to meet each other.

Therefore, we can substitute $x_j * x_k$ part in 3.4 with the number of corresponding edges currently exist in the lattice, namely each exiting edge for a specific reaction will contribute 1.0 for this part. Hereby the term "corresponding edges", we mean lattice edges that connect exactly two reactants of a specific reaction. To real with the case that two different reactions have identical reactants, the relevant edge will contribute evenly for each reaction with a sum of 1.0, for example, if two reactions share the same reactants, then the corresponding edge will contribute 0.5 for each reaction's propensity function in this part. To this end, we complete the problems about how to calculate the propensity functions related to those three kinds of reactions that can happen in the lattice model. Thus, with those functions, we can already determine which reaction will happen and how long the waiting time is to follow the traditional SSA. We can directly update our time varies according to the resulting waiting time. We will then take care of how to use those results to update the associated lattice configuration. As we are studying in the stochastic model here, which means we only need to randomly find a corresponding structure in our system with correct reactants (or reactant). By corresponding structures, we mean that single-reactant reactions can be thought of as sites and two-reactants reactions are edges. The "external source" reaction actually disappears for any lattice model with occupation restriction as one can always view each unfilled space in one site as an inert particle and then change all the "external source" reactions into a reaction with non-zero reactant according to its products. No occupation restriction case is even easier as one can just randomly pick a site or sites according to specific chemical model, which is not within the scope of this paper. As long as we find the corresponding structure, we can then update the lattice configuration according to our definition of the associated reaction. After we complete the configuration update, we need to update the propensity function again. The most straightforward method is to re-scan the whole system and account for different kinds of sites and edges again. However, as only those sites change their states and the edges connected to those sites become different from

old configuration, instead of re-scanning the whole system again, we can simply modify the propensity functions according to the information we obtain through clearly recording the states of those affected sites and edges before and after the configuration updating. After we fulfill this work, we complete the loop and can then iterate this process.

To this end, we have finished our algorithm iteration construction. In summary, under all our assumptions in the previous section about our general formulation of the chemical systems we want to simulate and assumed that we already have the initial configuration of our system, the Generalized SSA algorithm for stochastic lattice model proposed by us can be described as follows:

- *Step.* 0(initialization) Initialize the time variable $t = 0.0$.
- *Step.* 1 Calculate or modify the propensity functions for all the possible preset reactions according to current lattice configuration and our construction above.
- *Step.* 2 Simulate as the traditional SSA, namely obtain the reaction index and the time increment, update the time variable t immediately.
- *Step.* 3 Randomly searching in the lattice, find the correct structures and then update the lattice configuration according to associated reaction rule. Then jump to *Step.* 1 until reach the time wall.

Again, if we utilize exactly the same simplest example, we propose in section 3.3 about the single species A, as well as the same initial configuration as Fig. 3.1. Our simulation will process as follows.

After our system has been initialized, we will calculate the propensity functions for each reaction first. Here for our preset reactions, we know that an edge of $A - \varnothing$ will contribute to both the second and third reactions there, and each site occupied by a particle A will

benefit the first one. We can simply account for each type of these structures then. We have 12 edge $A - \varnothing$ and 4 sites occupied by a particle A, thus we can then calculate our propensity functions for each reaction as follows:

Reaction	Propensity
$A \xrightarrow{a_1} \varnothing$	$a_1 * 4.0$
$A + \varnothing \xrightarrow{a_2} A + A$	$a_2 * 12.0$
$A + \varnothing \xrightarrow{a_3} \varnothing + A$	$a_3 * 12.0$

With those propensity functions, we can then begin our simulation as original SSA first, employing the simplest Direct method. We use two random numbers to calculate the waiting time as well as determine which reaction happens. We can then update our time variable t according to the waiting time. Suppose the reproduction reaction (the second one in the set. 3.2) is selected, we randomly search all the edges (if the first reaction is selected, we need to search all the sites) in our lattice until we find a $A - \varnothing$. According to our reaction formulation, we then switch this $A - \varnothing$ edge to a $A - A$ edge, namely let an additional particle A occupy the empty site here. Thus we have completed the configuration update step. To modify the propensity function correctly, we need to monitor the configuration change carefully for both sites and edges. For example, if the $A - \varnothing$ randomly selected by us is shown as Fig. 3.3 as the red one. A particle A will then occupy the empty site at its right end. Thus, for the reaction set we considered above, we will have an additional particle A appear, we need to modify the propensity function of the first (spontaneous death) reaction by increasing the x_j factor in the second equation in Eq. 3.4 by 1.0. Furthermore, we will have two $A - \varnothing$ edges disappear, while two new $A - \varnothing$ edges appear, thus we don't need to modify the other two propensity functions. Otherwise, we will also need to modify the other two according to the net change in the number of $A - \varnothing$ edges. After we finish updating

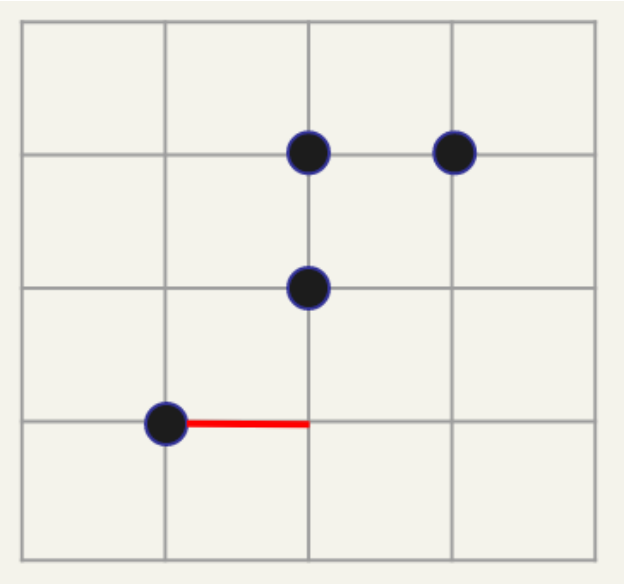

Figure 3.3: An example 5×5 square lattice with 4 sites are occupied by particle A (black dots), the red edge is selected to have a reproduction reaction happen

those propensity functions, we make a loop and can just go back to the standard SSA steps and iterate everything successively, until we reach the preset time wall. That is an explicit example of our own proposed generalized SSA for the stochastic lattice model.

3.4.2 Modification version

A crucial problem of the algorithm we proposed above is that, which looks similar to the StochSim case for those stochastic lattice model or even worse, after we determine which reaction is supposed to be on fire through the standard SSA, we then need to randomly search our lattice until we found a corresponding site or edge that occupied by or connecting the correct reactant(s). This step can be vastly cost when the corresponding sites or edges are rare in a current lattice configuration, just like in the StochSim case when active particles are much less than empty sites. Similarly, we can solve this problem by constructing some data structures that allow us to do random searching among the correct reactions directly.

Specifically speaking, we need data structures for each reaction individually, which allows us to do a random selection for a specific reaction and then find the corresponding structures (edge, or site) in our lattice directly. We can define those data structures as a representation of our system in the "reaction-based" space. Therefore, in other words, we need a complete mapping from the "reaction-based" space to the "particle-based" space, namely the lattice configuration. However, to keep those two structures consistent with each other, we also need an inversely mapping to modify those data structures in "reaction-based" space correctly, which makes the algorithm here more complicate.

Let us consider the simple example we have used many times here. To construct the completely double directions mapping between the "reaction-based" space and "particle-based" space there, in additional to the lattice configuration information, the propensity functions, we also need two arrays that recording the position of all the sites occupied by a particle A and all the edges that are of $A - \varnothing$ type. Furthermore, two inversely mapping structures are also needed to find out whether the associated site or edge is corresponding to a reaction and if so where it is in its "reaction-based" data structures. The reason why we need the backward mapping is that after a reaction happens, the system configuration will also change, as well as its "reaction-based" space representation. To keep both representation correct, we should either re-scan the whole system to rewrite its corresponding data structures in "reaction-based" space or correctly modify them. We choose the latter to avoid unnecessary costs. However, this choice will obviously increase the time cost anyway as we need additional operations to modify those data structures in "reaction-based" space. The programming complexity will increase as well.

3.5 Complexity analysis and comparison

In the last section of this chapter, we present a detailed analysis of the computational costs of our own proposed generalized SSA for stochastic lattice model, as well as for the corresponding generalized StochSim algorithm we have discussed in the motivation section. We will compare those two algorithms which can be used for the same systems. We will present a further implementation for the more specific model in the next chapter.

Let us first assume that we are working with the optimized version for either the generalized SSA or StochSim then, namely to avoid the multiple randomly searching steps we construct the mapping structures for each one. However, the mapping structures are relatively simple for the generalized StochSim compared with our generalized SSA case. We jump into the system initialization steps time cost and just assume they are the same for both algorithms, although they are actually different from those algorithms as for our generalized SSA case we need to initialize more complicated data structures. We will mainly focus on the time consuming of each loop then.

We can then write the time consuming of one generalized SSA algorithm as:

$$C_{SSA} = 3 * C_{ran} + C_{fet} + C_{cal} + C_{a0} + C_r + C_{ope} \tag{3.5}$$

Here C_{ran} means the cost of generating one random number; C_{fet} denote the cost of address fetching process in the mapping from one space to another for our generalized SSA algorithm; C_{cal} is the additional math cost for SSA, where we need to calculate the sum of propensity functions as well as a logarithm operation to generate the waiting time; cost C_{a0} stands for the operation to find the next on fire reaction; C_r is the cost of modifying the system configuration when a reaction happen; C_{ope} then represent the cost for correctly modify the mapping data structures here.

Similarly, we can also know the cost for each successfully reacting loop for the generalized StochSim algorithm:

$$C_{StochSim} = 3 * C_{ran} + C'_{fet} + C_r + C'_{ope} \tag{3.6}$$

Here we also need to generate three random numbers to complete successfully reacted loop. We may reduce one random by also constructing a completely mapping on from the "reaction-based" space to "particle-based" space and inversely, which is obviously unnecessary for the generalized StochSim case as it is intrinsically "object-oriented" or "particle-based". We will not consider this case more in our study then.

Therefore, by comparing Eq. 3.5 and Eq. 3.6 we can clearly obtain that:

$$C_{SSA} - C_{StochSim} = (C_{fet} - C'_{fet}) + C_{cal} + C_{a0} + (C_{ope} - C'_{ope}) \tag{3.7}$$

In the following, we will then consider each term above individually.

As we can see that the first term above which has been grouped in a bracket is the address fetching cost difference between those two algorithms. In the generalized StochSim algorithm discussed above, we usually need to do address fetching two or three times in each successfully reacting loop, one happens when we randomly find our first reactant; one or two may happen after we have changed the system's configuration according to the reaction rules and then modified the active particle list to keep consistent with our system. In the generalized SSA, we may need at most ten times: one time when we randomly find the specific reaction structure in our system to happen after we use traditional SSA determines the reaction type and the rest happen after we vary the system state according to the reaction rules (at the extreme case both reactants are changed, thus we have two sites and seven edges may change, together give us nine). In other words, $C_{fet} \approx 3 * C'_{fet}$. The situation will be similar for C_{ope} and C'_{ope}. We can then discuss C_{cal} and C_{a0} next. Clearly, C_{cal} will

include at least M times ADD operations to calculate the sum of all propensity functions and one logarithm operation. However, in the worst case, C_{a0} will contain $M-1$ ADD operations and the same amount of COM operations. Thus, for chemical reaction and diffusion systems with too many reactions or even too many active particle species, those two terms can be the dominate cost here. However, even for those systems that are not bearing those $O(M)$ complexity very much, our generalized SSA will still be more costing than generalized StochSim, as either C_{fet} or C_{ope} is about two times more costing than C'_{fet} or C'_{ope}. However, one advantage of generalized SSA is that, for each loop, we will always have a successful reaction, which is not the case in the generalized StochSim algorithm. Thus we may need also to consider the cost generated by those unsuccessful reactions, which will eventually increase the first two terms in Eq. 3.6. The specific amount of additional cost will depend on the specific system set up, which is out of the scope we can generally talk about here.

In conclusion, our generalized SSA for stochastic lattice model is supposed to be more cost than the original generalized StochSim algorithm, especially for chemical systems with too many reaction types and/or too many active species. However, as the advantages of the original and thus our generalized SSA is significant for some systems, it is still valuable to explore our algorithm more.

Chapter 4

Numerical Results and Discussion

4.1 Introduction

In this section, we will apply both the widely studied generalized StochSim algorithm and our own proposed generalized SSA for some specific models to prove the correctness of our algorithm and compare the efficient as well as the advantages or disadvantages of them. We use a three species cyclic predator-prey model, namely the May-Leonard model and an Irreversible Surface-Reaction model as two examples to test our generalized SSA. The cyclic predator-prey model can be used to describe biodiversity in ecology and biology. It helps us realize the co-evolutionary dynamics of three coexisting species in cyclic competition, such as the coexistence of three strains of E. Coli in bacteria experiment. The interesting spatial pattern will also emerge in this kind of system. On another hand, the kinetic surface-reaction model is about the reaction of carbon monoxide and oxygen on a catalyst surface. This model was first investigated by R. M. Ziff, E. Gulari and Y. Barshad. Thus it is also called the ZGB model. It is thought to exhibit interesting steady-state non-equilibrium behavior and two types of phase transition that can be viewed experimentally. Both of those two stochas-

tic lattice models have been simulated using the generalized StochSim and widely studied. However, we will prove that our new proposed generalized SSA can provide us with similar results that demonstrate the correctness of our algorithm. Details about each model will be discussed in the following section. We will investigate the May-Leonard model in section 4.2 and then the ZGB model in the next section; finally, we will make our conclusions.

4.2 The May-Leonard model

We will focus ourselves on an interesting simulation model that is called a two-dimensional May-Leonard model in this section. We will implement this kind of system through both the generalized StochSim algorithm as well as our generalized SSA. We compare those results generated by both of those two methods to prove the correctness of our new algorithm and to explore their common points and differences in their dynamic features. We will first describe the general information about this model first.

4.2.1 Model description

The rock-paper-scissors system is a non-trivial simulation model to study the biodiversity in ecology and biology. It includes three-species who are cyclically preying each other like the well-known rock-paper-scissors game: 'rock' smash 'scissors', 'scissors' cut 'paper', 'paper' wrap 'rock'. This model and its variants can be used to mathematically describe the co-evolutionary dynamics of three competing species. Here we are interested in a particular one that is introduced by May and Leonard [53]. There will be novel results with interesting spatial patterns emerging in this model when considered in a two-dimensional case which

makes important implications. Therefore it is widely studied during the last two decades. We first simply describe this model as follows: we let all particles live on a 2D square lattice, with each lattice site occupied with at most one individual. In other words, this is the only single-particle occupation restriction mentioned in the previous chapter. We can then treat the empty sites as passive particles. Therefore, we can apply our generalized SSA, as well as the StochSim algorithm for stochastic lattice model here. We label those three active species A, B and C, and the passive empty site $\varnothing$. According to the definition of May-Leonard model, the interaction between those three active species and the passive empty sites can be written as:

$$\begin{aligned} A + B &\rightarrow A + \varnothing, \ with\ rate\ \lambda; \\ B + C &\rightarrow B + \varnothing, \ with\ rate\ \lambda; \\ C + A &\rightarrow C + \varnothing, \ with\ rate\ \lambda; \\ X + \varnothing &\rightarrow X + X, \ with\ rate\ \mu; \end{aligned} \tag{4.1}$$

where X refers to any one of the three interacting species. We assume that all three active species have identical predation rates as well as the same offspring production rates so that there are no explicit advantages among them. This model violates the number of conservation of the traditional rock-paper-scissors model. Here as we assume the at most single-particle occupation restriction, on-site reaction assumption can never be applied. Thus, we also assume that all those reactions can happen only when two particles are nearest-neighbor of each other. Furthermore, in our 2D square lattice model, we also consider the exchange of the nearest-neighbor particle as well as the hopping process in this specific model:

$$\begin{aligned} X + Y &\rightarrow Y + X, \ with\ rate\ \epsilon; \\ X + \varnothing &\rightarrow \varnothing + X, \ with\ rate\ D; \end{aligned} \tag{4.2}$$

Here again, X, Y can be either A, B, C. However, we will not consider much about the exchange between the same species here. However, for simplification and following the previous references [54, 54, 55] assumption, where they actually do not separate the hopping and exchanging processes, we can just set $\epsilon = D$. Detail effect about this simplification is out of the scope of this study.

With those reactions defined in (4.1, 4.2), as well as all those assumptions we make above, we can implement this four-state stochastic RPS model on two-dimensional lattice. We use 256×256 or 512×512 grid size for each system if not claim this parameter explicitly. We also simply choose periodic boundary condition and randomly initial configuration with equal initial density for each site state, namely $a(0) = b(0) = c(0) = e(0) = 1/4$. Here, $a(t), b(t), c(t)$ are density function for sites occupied by three active particles and $e(t)$ is for empty sites. With all those numerical setups we can then simulate systems. We mainly choose out reaction rates in simulations like: $\lambda = 1.0; \mu = 1.0; D = \epsilon = 5.0$. However, in the StochSim algorithm, things that really matter are the ratios between each reaction probability. We will then generate simulation results using both the generalized StochSim algorithm and SSA. We will then compare those results to demonstrate the correctness of our new proposed generalized SSA as well as test the performance of different stochastic lattice model simulation algorithm.

4.2.2 Spatial pattern comparison

This stochastic spatially extended three species model had first been studied by May and Leonard on rate equations level [53] for well-mixed systems with a large number of individuals. In the deterministic description, three absorbing fixed points as well as a coexisting fixed point where all three species survive. The reactive fixed point is thought to be unstable and

under the finite individual number situation, the existing fluctuations will make the system finally reach one of the absorbing states with only one species fills the whole system. Due to the symmetry reactions between all three species as we assumed earlier, each subpopulation have an equal chance to survive and final results will be strongly affected by the initial configuration.

However, we are more interested in the spatial extended stochastic lattice model case as we have discussed its description in the previous subsection with cyclical interaction as well as the hopping and exchanging process. We will employ this approach to construct our simulation code. As demonstrated and investigated by various studies [53–58], there will be a kind of noisy, regular, geometric spiral wave patterns emerging in system configuration on the scale of a large enough system size and with appropriate exchanging and hopping rate selection. The emergence of this wave pattern is a feature shared by many complex systems that are supposed to be out-of-equilibrium across multiple disciplines, such as the Belousov-Zhabotinsky reaction [59]. It can be also studied by mapping and recasting the stochastic and deterministic part of dynamics here in the form of a complex Ginzburg-Landau equation(CGLE) with noise term [55, 60], which provides us with a theoretical explanation about the emergence of spiral wave structures as this is also an important feature of 2D CGL system, if one numerically solves it on 2D lattice [61].

However, in all those lattice model simulation studies mentioned above, the algorithm they employed is the generalized StochSim algorithm for the stochastic lattice model. Thus we will then ask that what if one simulates the same stochastic spatially extended three species model using the generalized SSA we proposed in the previous section. To answer this question, we generate both simulation codes using both of these two algorithms and then compare them carefully from various aspects. We will first compare them from the phenomenological aspect.

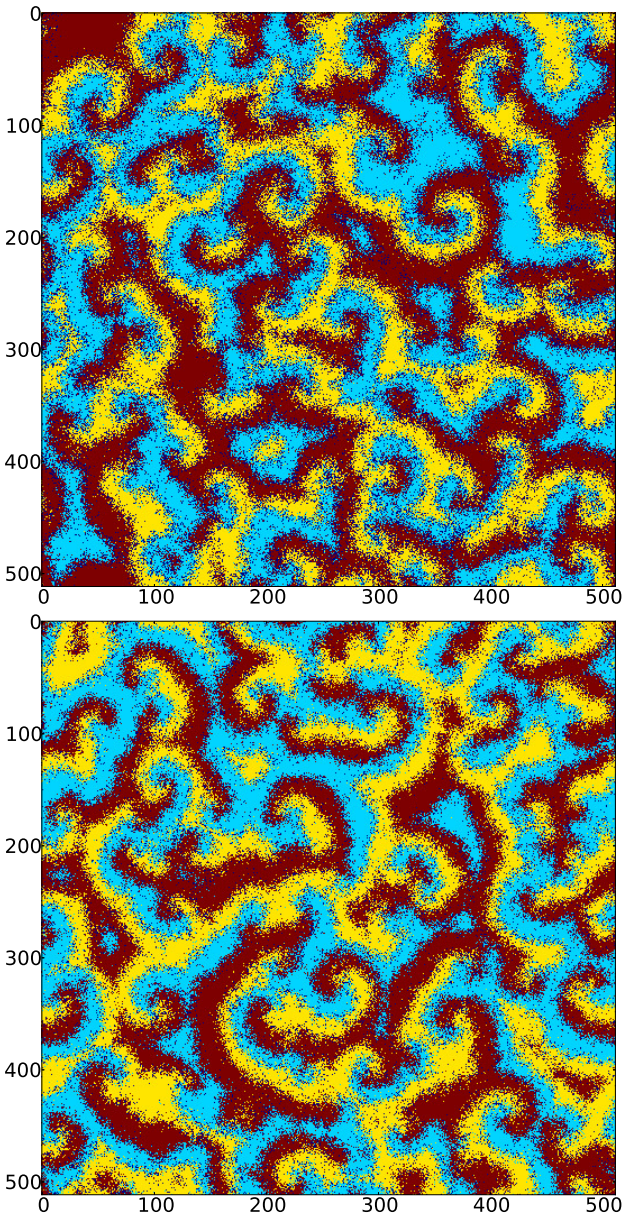

Figure 4.1: Snapshots of the spatial particle distribution of systems simulated using the generalized StochSim(upper) and SSA(lower). Different colors stand for sites occupied by three active species (yellow, red and light blue) or left empty (dark blue). Each system has a size of 512×512 and a total random initial configuration with equal initial density for all four site states. The rate values are implemented $\lambda = 1.0; \mu = 1.0; D = \epsilon = 5.0$. The time when taking these snapshots is $t = 3000MCS$ for the StochSim case and $t = 1000.0$ for SSA.

In Fig. 4.1, we plot a typical snapshot for May-Leonard model systems that are simulated using different algorithms. Here we can see all three active species coexist and clear spiral structures can be seen in each system. One can also compare those two system configurations generated using a stochastic simulation method and random initial and make the conclusion that there is no significant different feature between those two systems. However, to enhance this conclusion. We need to compare those systems generated through different algorithm quantitatively.

Before we move to the detail quantitative comparison section, it may be also valuable to mention another interesting difference between those two algorithms. As we had discussed in the previous section, we claimed that the original SSA and hence our generalized algorithm will provide us results that are more exact if one wants to describe those systems using the chemical master equations. On the other hand, if one chooses a very small time step for the StochSim algorithm, it can be viewed as a first-order approximation to the corresponding SSA. The difference between those two algorithms will be more significant if one considers reactions with rates of different magnitudes. This will be true for our May-Leonard model with hopping and exchanging processes as the reaction rates of those two processes are usually considered to very large than those of interaction or reproduction reactions. Furthermore, according to previous study[54–57], when the mobility of particles, which is defined as proportional to the exchanging and hopping rates, is increased, the spiral structures will also grow in size and disappear for large enough mobility for finite system size as they are even larger than the entire lattice. In the spirals absent case, the coexisting state will also break down and the system will perform a uniform state where only one species is surviving, namely an absorbing state. It is easy to understand this transition as when one keeps increasing the mobility of active species, a spatially extended system will behave like a well-mixed case gradually. However, those conclusions are obtained when one employs the generalized StochSim algorithm, what if one change their method to our generalized SSA?

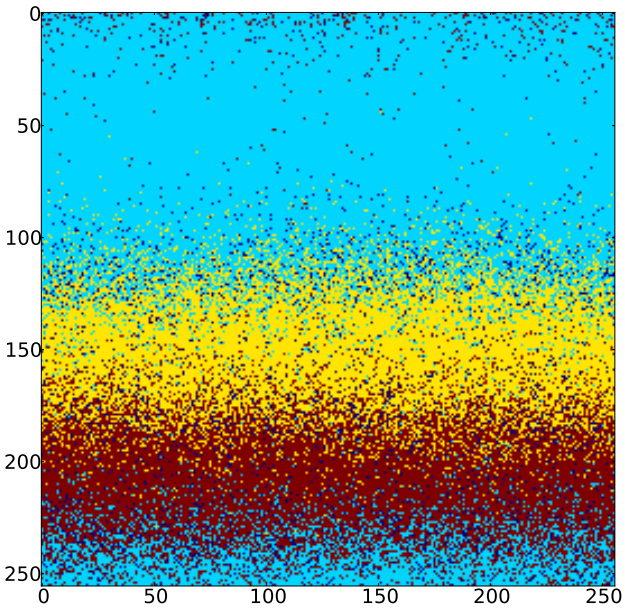

Figure 4.2: Snapshots of the spatial particle distribution of systems simulated using the generalized SSA(right). Different colors stand for sites occupied by three active species (yellow, red and light blue) or left empty (dark blue). Each system has a size of 256×256 and a total random initial configuration with equal initial density for all four site states. The rate values are implemented $\lambda = 0.5; \mu = 1.0; D = \epsilon = 75.0$. The time when taking these snapshots IS $t = 1000.0$.

Another example snapshot of system simulated using our new algorithm is presented in Fig. 4.2. Here a different set of reaction rates are using and diffusion is assumed to be much larger than other reactions(150 times than species interaction and 75 times than reproduction). Here, instead of directly transit from spiral structures existing state to a uniform state, an intermediate state is observed. Their planar wave-like structures emerge in system configuration when all spirals with different winding directions annihilate with each other. This phenomenon is also seen previously by Mobilia et. al. [62]. They combined the May-Leonard model as we used in our implementation as well as other additional reactions, for example, species mutations which mimic the fact that side-blotched lizards Uta stansburiana undergo throatcolor transformations [63] or phenotypic switching of uropathic E. coli [64].

They also used a so-called 'meta-population model' which is exactly the Gillespie algorithm for mesoscopic reaction-diffusion systems. They concluded that this phenomenon happens when the system size is of the order of the wavelength of the spiral waves. Those planar waves like structures then look like the arms of the original spiral waves. Or more generally speaking, this is a finite-size limit that can only be obtained by SSA like algorithms. In other words, this limit case will be absent if one does simulations using algorithms with StochSim concepts since they are only a first-order approximation of the corresponding SSAs. This is a significant advantage of our new proposed algorithm. More tests are definitely needed to check if there are other useful findings associated with this difference between the original generalized StochSim and our new proposed SSA.

4.2.3 Quantitative comparison

To quantitatively compare those May-Leonard systems generated using the same reaction parameters but different simulation algorithms, we need quantities to characterize the emerging spatial structures in our system as well as depict some associated temporal features. Follow methods that were used in previous work [57], we will employ the equal-time two-point correlation functions of the same or different species to describe the spatial features of our systems in the quasi-steady case. The formula to compute those functions are shown as follows:

$$\begin{aligned} C_{AA}(x,t) &= \langle n_A(j+x)n_A(x)\rangle - a(t)^2 \\ C_{AB}(x,t) &= \langle n_A(j+x)n_B(x)\rangle - a(t)b(t) \end{aligned} \tag{4.3}$$

Here $n_A(j,t)$ is the local population number of species A at lattice site j, $a(t)$ is its spatially averaged population density $a(t) = \frac{1}{N}\sum_j n_A(j,t)$. Things is similar for species B. We also calculate the Fourier transformation of $a(t)$ to characterize the temporal feature of our system.

$$a(f) = \int a(t)e^{2\pi i f t}dt \tag{4.4}$$

The reason why we can only consider species A (although when we calculate $C_{AB}(t)$ we need information of species B) is because of the underlying symmetry among the species A, B and C. We will also do averaging between independent systems to rule out fluctuations generated from the random initial configuration and the stochastic processes.

However, before actually compare results extracted from systems simulated using different algorithms, there is another difficulty existing. As it is clearly seen from the algorithm details which have been introduced in the previous section, the time scale of those two algorithms are not consistent with each other. Specifically, we use the so-called Monte Carlo step (MCS) to scale the time axis in the generalized StochSim algorithm. On the other hand, a continuous-time variable is employed in the SSA case. We should first rescale those two-time variables to synchronize them. However, since we are doing stochastic simulation, this objective is hard to obtain. A reasonable choice is to rescale the time variable of one kind of system (thus the frequency f after doing the Fourier transformation) to overlap the most significant peak value in its frequency spectrum. But this method will render the comparison using $|a(f)|$ meaningless. Therefore, we need other quantities to extract this re-scale factor. We decide to use the cumulative number of $N_i(t)$ to fulfill this job. We define $N_i(t_0)$ to be the number of how many times has reaction i happened from $t = 0$ to $t = t_0$. We account for this number for different reactions for each case and plot them respect to t. We will rescale the time variable t of one algorithm (we choose StochSim here) to collapse those curves generated from the different algorithms. For the specific reaction rates set $\lambda = 1.0; \mu = 1.0; D = \epsilon = 5.0$, a

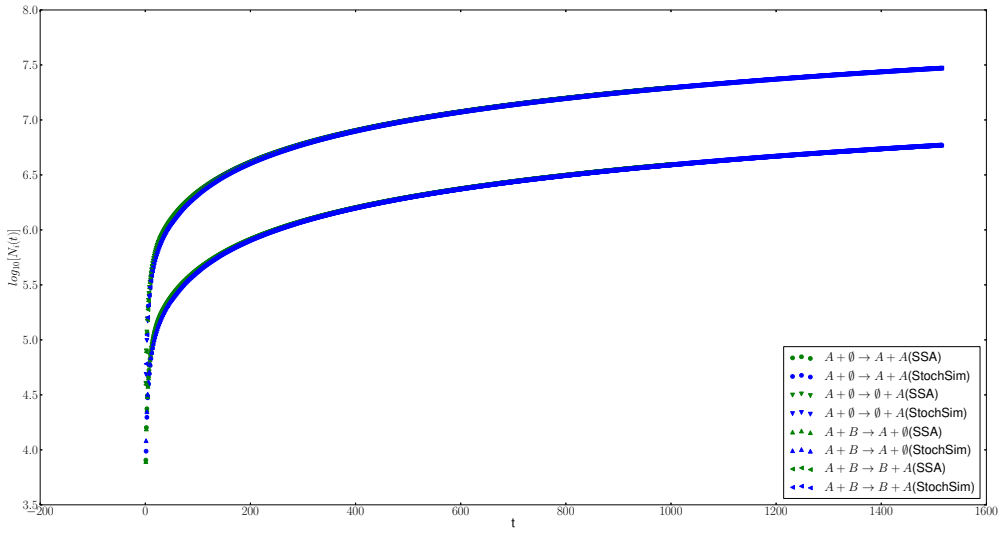

Figure 4.3: Cumulative reaction count curves for different reactions that are listed in their labels. The time variables of those generated from StochSim algorithm are rescaled according to a uniform factor $rs = 13.3$ to make them overlap with those simulated using our generalized SSA. Those curves are averaged over 100 simulation runs. Lattice size is 256×256

rescaling factor $rs = 13.2$ will perfectly make those curves overlap with each other. Fig. 4.3 shows the relevant curves.

With this factor, we can then successively compare $|a(f)|$ computed from systems that using different algorithms, as well as the equal-time correlation functions $C_{AA}(x,t)$ and $C_{AB}(x,t)$.

As one can directly see in Fig. 4.4 that, after we rescale the time axis for those systems simulated using the generalized StochSim algorithm, both $C_{AA}(x)$ and $C_{AB}(x)$ are consistent with each other. However, three reasons may be associated with a slight difference between black and red curves. First of all, we have stochastic noise inside each system, which we can not totally rule out through statistic average. Secondly, as eventually systems actually stay in a quasi-steady state and the time when we measure $C_{AA}(x)$ and $C_{AB}(x)$ are not the same for two different algorithms even after the rescaling, there are supposed to be a slight deviation. A third reason is that, as claimed by a previous study, the StochSim algorithm is

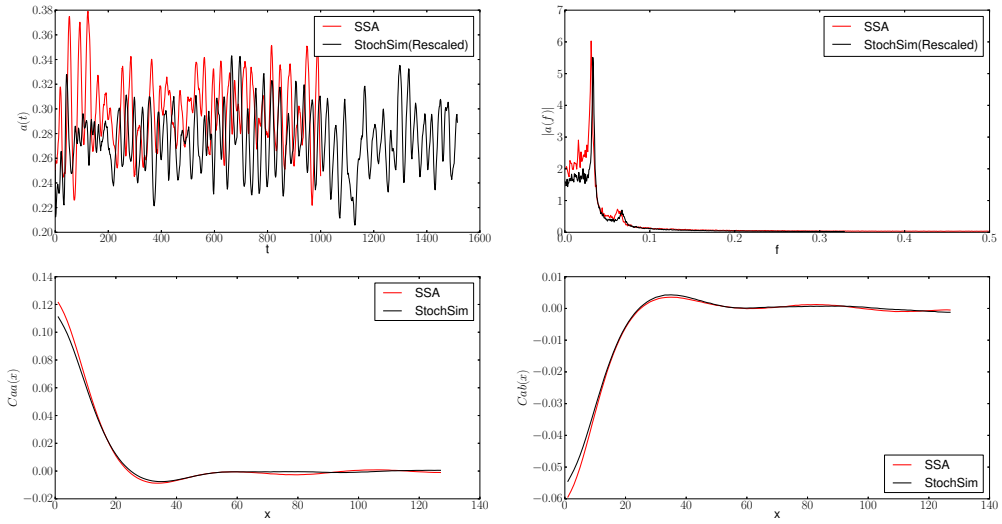

Figure 4.4: Quantitative observables for stochastic May-Leonard system on 2D lattice with 256×256. $|a(f)|$(upper right), $C_{AA}(x)$(lower left) and $C_{AB}(x)$ are averaged over 100 simulation runs. $a(t)$ is only an example generated from one single run. The reaction rates are $\lambda = 1.0; \mu = 1.0; D = \epsilon = 5.0$. $C_{AA}(x)$ and $C_{AB}(x)$ are obtained at $t = 1000.0$ for those SSA systems and $t = 20000MCS$ for the StochSim case.

just a first-order approximation to the corresponding SSA. Aside from those spatial feature comparisons, they also share similar temporal features as shown in the $|a(f)|$ curves. They roughly have the same shape as well as the same peak frequency, which demonstrate the consistency of those two algorithms, at least for this specific May-Leonard model.

4.2.4 Efficiency comparison

We will try to compare the efficiency of those two algorithms when applying them to our May-Leonard model. We will measure the CPU cost for each algorithm. We will use different reaction rate sets, as we notice that different from the StochSim algorithm, SSA is actually very sensitive to the reaction rates, especially the largest one. Thus, our scenario is as follows: we will mainly vary the largest parameter, namely the exchanging and hopping

Rate values	Rescaling factor	SSA CPU cost(s)	StochSim CPU cost(s)
$\lambda = 1.0; \mu = 1.0; D = \epsilon = 5.0$	13.3	148.30 ± 0.52	129.30 ± 0.93
$\lambda = 1.0; \mu = 1.0; D = \epsilon = 10.0$	26.3	278.16 ± 2.27	250.35 ± 1.76
$\lambda = 1.0; \mu = 1.0; D = \epsilon = 20.0$	N/A	419.91 ± 2.31	N/A
$\lambda = 1.0; \mu = 0.5; D = \epsilon = 5.0$	15.3	164.74 ± 0.67	127.81 ± 1.30
$\lambda = 1.0; \mu = 0.5; D = \epsilon = 10.0$	30.3	311.15 ± 2.78	248.04 ± 2.30
$\lambda = 1.0; \mu = 0.5; D = \epsilon = 20.0$	61.3	555.03 ± 12.51	503.36 ± 5.05

Table 4.1: The overall CPU costs of 100 simulations for SSA and StochSim simulated May-Leonard model system on 2D lattice

rate. We will also slightly vary other parameters a little bit to obtain a better understanding of this question. Our simulated system is still 256×256, and other numerical setups are identical as introduced in the previous subsection. We do statistics over 100 independent different realizations. We will also use the extract uniform scale factor to determine the pseudo time of the SSA simulation and the MCSs used in the StochSim algorithm. For example, if we have a factor 13.2, as shown in Figure 4.3 and those SSA simulated systems use pseudo time $t = 1000.0$ as time wall, then we will simulate those StochSim systems for $13.2 \times 1000.0 = 13200$ MCSs, to match their evolutionary history.

Here, we fix the time wall of those SSA systems to be $t = 1000.0$ and only vary the MCSs simulated in the StochSim case. The N/A means those systems simulated using StochSim are quickly falling into the absorbing state where the whole system is occupied by a single species, while in SSA case we can still obtain active systems. Besides this special case, from the third and fourth columns in Table 4.1, we can clearly see that the CPU cost of SSA is always higher than the StochSim algorithm, but they are still comparable. Furthermore, we can also find that those differences are becoming smaller as the scaling factor get larger, or in other words when the diffusion related reaction rate get larger. This fact is consistent with our conclusion that SSA will perform better than those models with reaction rates of a different order. However, limited by the extinction properties of the May-Leonard model studied here, we can not further increase ϵ here as those StochSim simulated systems and

even SSA systems may transit to absorbing state. We will try to eliminate this restriction and check our prediction that our SSA method may perform even better than StochSim algorithm when the order difference becomes even larger, which will be tentatively done in the next section by implementing another model.

4.3 Ziff-Gulari-Barshad Model

4.3.1 Model Description

In addition to the May-Leonard model which is discussed in the previous section, we also test our algorithm using another many-body system that can be simulated using Monte Carlo simulations on a simple square two-dimensional lattice of active sites and single-occupation restriction, the Ziff-Gulari-Barshad(ZGB) model or $A-B_2$ model. It is an irreversible kinetic surface-reaction model considering the reaction of carbon monoxide and oxygen atom on a catalyst surface. It is first proposed by Ziff et al. in 1986, thus it is known as the ZGB model. Here the catalytic surface is represented by a two-dimensional square lattice. Three basic steps are usually assumed in heterogeneous catalytic chemical reactions. First of all, those reaction particles are adsorbed on the surface; secondly, they react with each other; the final step is the desorption of their products. The final step here is also known as the Langmuir-Hinschelwood process[65]. It ensures the regeneration of the catalyst.

In the simulation model considered by us here, the ZGB model, detail reactions are estab-

lished by the following steps:

$$\begin{aligned} CO(gas) + V &\rightarrow CO(ads),\ with\ rate\ y_{CO} \\ O_2(gas) + V &\rightarrow 2O(ads),\ with\ rate\ y_O \\ CO(ads) + O(ads) &\rightarrow CO_2(gas) + V,\ with\ rate\ y_{CO_2} \end{aligned} \tag{4.5}$$

Here, (gas) means a gas-phase molecule of CO or O_2; (ads) indicates that the molecule is adsorbed on the catalysis surface; and V means an empty active site. To be more specific, those irreversible surface reacting processes can be sum as follows. First of all, the catalyst surface, which is simplified into a square two-dimensional lattice with active sites. Each active site can be occupied by only one adsorbed particle. This catalyst surface is exposed to an environment filled with gas-phase molecules of CO or O_2. Both kinds of these molecules can collide with blank sites directly can then absorb immediately. O_2 will dissociate into two O atoms and occupy two separate empty sites, which are the nearest neighbors of each other. On another hand, CO molecule needs only a single site to be adsorbed. They can be viewed as the first step mentioned in the general heterogeneous catalytic reaction process. Besides those two absorbing reaction, an adsorbed state CO can react with an adsorbed O atom on one of its adjacent sites (in the two-dimensional cases, it has four nearest neighbor) and generate a CO_2 molecule, which will desorb instantly and restore two empty active sites on the catalyst surface, complete the second and third standard step mentioned above. Those desorbed CO_2 will never interact further with our catalyst surface again. In other words, we assume that our environment has a very large volume and its filling gas is continually replenished by a fresh feed[66], so that the composition in the gas phase will never be changed by either the desorption of CO_2 or absorption of CO and O_2.

Many other mechanisms are not considered in this simplified model, such as the diffusion processes, desorption of CO or reassociation and desorption of two adjacent O atoms, which

may be important for further studies. However, in the earliest version first proposed by Ziff et al. [66], they even not consider the reaction rates of those three standard reactions in (4.5) to avoid using any energy parameter here and leave with only one control parameter. They assumed that the three reactions in (4.5), which produce desorbed CO_2, happens immediately when two reactants occupy adjacent sites. This simplification implies an infinite reaction rate which very idealized and can not be implemented through SSA, where reaction rates are explicitly used. We need to improve this most simplified model a little bit by explicitly define reaction rates for all those reactions in (4.5).

Similarly as the May-Leonard case, almost all those previous studies investigating this ZGB model or its modified version employing the StochSim algorithm for stochastic lattice model as discussed in the previous chapter. We then implement our simple improved ZGB model with both the SSA and StochSim algorithm as what we did in the previous section for the May-Leonard model. We will first demonstrate that our SSA method can provide us similar results as the widely used StochSim algorithm. After that, the related efficient of each algorithm will be compared.

4.3.2 Correctness checking

To check the consistency between those systems simulated using two different algorithms, we need to first introduce its general behaviors that have been studied by previous work [66]. There exists a typical steady-state of this catalyst reaction system, where the system is reactive. Both CO molecules and O atoms coexist on the catalyst surface and some of those lattice sites also stay empty. There, there will be continuously absorption reactions of O_2 and CO happen, which will generate new adjacent CO and O pairs. Those pairs will then react soon, as the third reaction in (4.5) is usually assumed to have a much larger reaction

rate than the other two (that is why in the most original models, they even assumed infinite reaction rate for this reaction). After this reaction happen and produced CO_2 then desorbs, two active sites are vacated, which completes the loop and systems can keep repeating those steps as we have the replenishment assumption.

Figure 4.5 illustrates two examples of system configuration simulated using different algorithms. We can clearly see that, in both plots, the active sites are mainly occupied by O atoms and CO clusters are rare. Actually in our case. those CO molecules tend to be isolated among vacated active site clusters, the latter commonly occur among O atoms. Those descriptions are consistent with each of those two plots, which means those two different algorithms are almost doing the same thing. The occupation percentage of adsorbed species are also similar to each other for those two examples. About 70% of the active sites are occupied by O atoms, only 1.6% are CO molecules and rest left blank.

If one fixes two of these reaction rates mentioned in Figure 4.5 and varies the last one, this reactive steady state can exist for some intervals. The two-phase transition will happen when one goes out of the interval on both sides. The only steady states outside of the interval are "poisoned" catalyst surface covered by either pure CO molecules or pure O atoms. For example, if we fix $y_{CO} = 1.0$ and $y_{CO_2} = 20.0$ and keep changing y_O, then only for $y_1 < y_O < y_2$ we have the reactive steady state for our ZGB systems. We measure y_1, y_2 using our SSA algorithm, which gives us $y_1 = 0.478 \pm 0.001, y_2 = 0.804 \pm 0.001$. However, the StochSim version only gives us probability ratio of those three reaction rates, if we fixed $y_{CO} : y_{CO_2} = 1 : 20$, and set $y_{CO_2} = 1.0$, the resulted $y_1 = 0.0239 \pm 0.0001$ and $y_2 = 0.0404 \pm 0.0002$, which agrees with those obtained from our SSA algorithm through a factor transformation by multiple the latter ratio result by 20.0.

Furthermore, those two transitions from the reactive steady-state to either CO or O "poi-

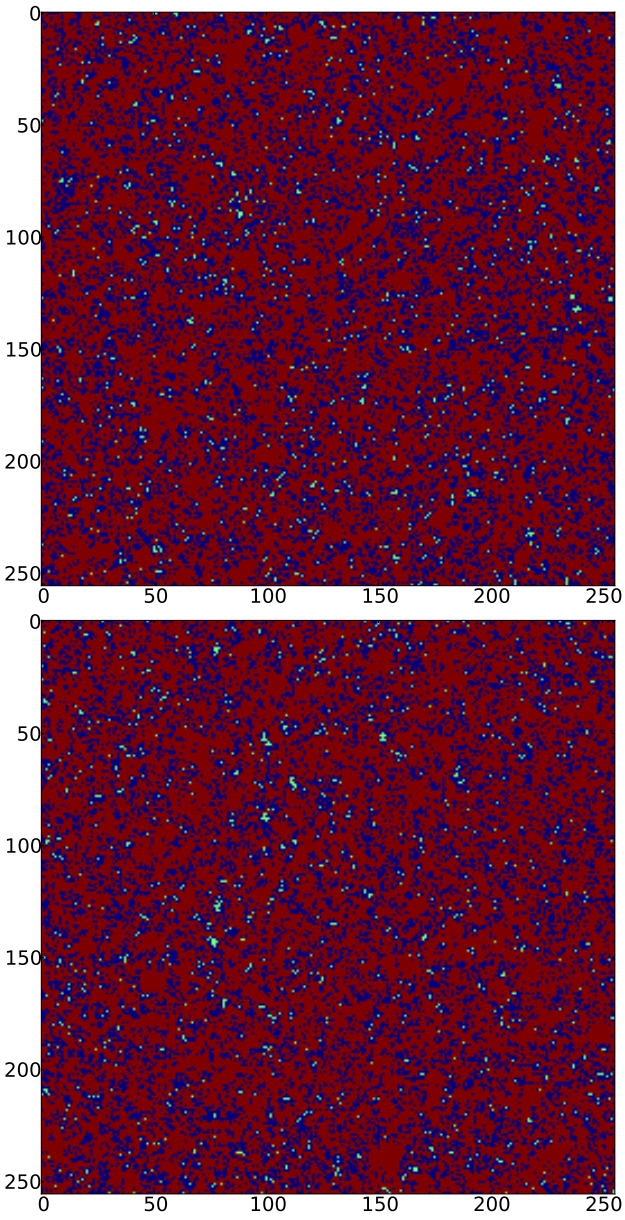

Figure 4.5: Snapshots of the spatial particle distribution of systems simulated using the generalized StochSim(left) and SSA(right). Different colors stand for sites occupied by CO molecule (light green), O (dark red) or left empty (dark blue). Each system has a size of 256×256 and a blank initial configuration with all active sites left empty. The rate values used in the SSA example are $y_{CO} = 1.0; y_O = 0.6; y_{CO_2} = 20.0$. Identical ratios between the reaction probabilities are employed in the corresponding StochSim simulated systems. The time when taking these snapshots is $t = 3000MCS$ for the StochSim case and $t = 1000.0$ for SSA.

soned" state are different from each other. The first one is a discontinuous or first-order transition, which can tell from the occupation percentage of either reactant species, for example exactly before the transition, the O atoms occupation percentage is about 44%, which is given by either algorithm. It will suddenly drop down to 0% after transit to the CO "poisoned" state. On another hand, before the O "poisoned" transition limit, the catalyst surface is almost fully covered by O atoms, again can be checked by both simulation versions. To this end, we believe we have enough pieces of evidence that our new proposed algorithm can simulate the stochastic lattice model similar to the traditional StochSim algorithm does. We will then compare their efficiency using this implementation as well.

4.3.3 Efficiency comparison

We will try to compare the efficiency of our SSA and the widely used StochSim algorithm with this ZGB model. A difference compared from the May-Leonard model we tested in the previous section here is that system will not fall into its absorbing state for some interval even if we increase the reaction rate of the most rapid reaction, namely the third one in (4.5) which is catalysis and generates CO_2, infinitely. As we have discussed and predicted, our new proposed SSA may even perform better than the original StochSim algorithm for models with a large gap between the most rapid and slowest reaction, which is limited by the transformation from a reactive state into absorbing state in the previous model, but not the case here. Therefore, we can further check this assumption.

We will compare the efficiency using the overall CPU cost, similar to what we did in the previous section. We will first measure the transforming factor between systems implemented using different algorithms by matching the numbers of successfully reacted reactions as a function of the pseudo time t or MCS in each algorithm, exactly the same as we did for

Rate values	Rescaling factor	SSA CPU cost(s)	StochSim CPU cost(s)
$y_{CO} = 1.0; y_O = 0.6; y_{CO_2} = 20.0$	30.0	248.24 ± 6.25	3282.83 ± 75.76
$y_{CO} = 1.0; y_O = 0.6; y_{CO_2} = 40.0$	60.0	324.98 ± 1.36	5553.55 ± 148.70
$y_{CO} = 1.0; y_O = 0.6; y_{CO_2} = 50.0$	75.0	255.32 ± 2.09	7180.21 ± 202.58

Table 4.2: The overall CPU costs of 100 simulations for SSA and StochSim simulated ZGB model system on 2D lattice

May-Leonard model again. The associated reaction rates of the first two reactions in (4.5) are fixed by $y_{CO} = 1.0; y_O = 0.6$ to ensure a reactive state for even infinite catalysis reaction rate, same as we used to show the example system configuration. The most rapid reaction rate is varied. On the other hand, in those StochSim codes, we fixed the CO_2 reaction probability by 100% and vary the other two, keep the same ratio as the reaction rates used in the corresponding SSA codes. 256×256 lattice size is employed again. We will also fix the time wall of the SSA codes by $t = 10000.0$ and then vary the limit of StochSim codes according to the transforming factor. Table 4.2 shows our measured CPU costs for codes implemented using different algorithms. Here, those systems simulated using the StochSim algorithm turns out to be much more costing than those using our SSA, as we tentatively predict in the previous section. The time needed here for StochSim codes is about at least one order larger than SSA codes. Furthermore, the rescaling factor that transforms pseudo time used in SSA to MCSs in the StochSim algorithm also shows a nice linear relation with the reaction rate of the most rapid or the catalysis reaction, namely the one we vary here. Two reasons give rise to this phenomenon. First of all, we only generate the scaling factor through rough observation, slightly change them will not make much difference. For example, if we change 30.0 to 31.0, those reaction account curves of codes implemented using two different algorithms will still roughly overlap with each other. Secondly, if looking at our reaction number curves carefully, we will find that the curve of the first reaction in (4.5) is approximately equal to that of the third one in (4.5) or the catalysis reaction. This behavior can be explained as follows: for this parameter choice, there are only quite a few CO exist

on the lattice while quite large O clusters exist, which can also be seen in Figure 4.5. As long as a CO absorbing happen, it will react with an adjacent oxygen atom soon. In other words, a CO_2 catalysis reaction will always follow an absorbing of CO. Therefore, the reactions that actually dominate the evolution of history should be those two absorbing reactions in (4.5). However, when we increase the reaction rate of the most rapid reaction in our SSA codes, respectively, we need to decrease the successfully reacting probability of those two in the StochSim algorithm case. This operation will proportionally reduce the successful reaction account at the average level at any specific MCS. On another hand, the SSA codes will never bear this issue. Thus, the time rescaling factor between those systems simulated using different algorithms will be asked for a roughly linear increase as one of them decreases inversely. The second reason here can also be used to explain why SSA here performs much better than the StochSim algorithm. As the most rapid catalysis reaction is restricted by the CO absorbing processes, the really dominate reactions happen in the evolutionary history are the first two in (4.5). However, limited by the fact that we can only set the most rapid reaction with a 100% successfully reacting probability and to keep the ratio of other probability as the corresponding SSA codes with reaction rates of different order, we have to give a very low successfully reacting probabilities for both absorbing reactions at each single trail. They can hardly happen then, which gives rise to the low efficiency of the StochSim algorithm when simulate the ZGB model here. However, in the May-Leonard model case, StochSim will not be bothered by this problem even when we increase D and ϵ to 20.0, because of the decoupling of the rapid diffusion processes (exchanging or hopping) and other two low reacting rates reactions.

4.4 Conclusion

To this point, we summarize our results and conclusions for this model test chapter. With the SSA for the stochastic lattice model as introduced by us in the previous chapter, we tried to implement two different stochastic lattice models using both this one and the originally existing StochSim algorithm. First of all, we demonstrated that our new proposed algorithm can be used to simulate those models and provide us results similar to those obtained for systems implemented using the StochSim algorithm, which means that our new proposed algorithm is correct. We also characterized and compared the efficiency of the SSA and StochSim algorithm using their CPU cost measured for simulating either the May-Leonard model or the ZGB model. We found that, in the May-Leonard case, our simulation results did not show the advantage of SSA. However, their time cost is still comparable. When further increased the reaction rates of the most rapid reactions, the SSA and StochSim cost time are approaching each other. We predict that if we can further increase this rate value, our SSA will perform better than the StochSim. The existing transition from a reactive state to an absorbing state when increase species' mobility prevents us from further check this prediction. A different wave pattern also emerged when we use the SSA, which can not be seen in the corresponding StochSim due to this transition happen much earlier in that case. The reason is that in a well-mixed case, the StochSim can be a first-order approximation of the relevant SSA, which is inherited by their spatial extended case. On the other hand, in the ZGB model case, the SSA turned out to perform much better than StochSim, as the high efficient most rapid reaction here is restricted by the lower efficient reaction with tiny successful reaction probability. More application and analysis of this new proposed algorithm will be needed in the future.

Chapter 5

Conclusion and Future Work

We presented one project in this thesis studying the simulation algorithm for the stochastic lattice model. Traditionally, the StochSim algorithm is used to simulate this particle-based spatially inhomogeneous model, as it has an object-oriented feature. However, to investigate these nonequilibrium systems more carefully, we want to drive our system from another approach. We tried to follow the concepts of another important stochastic simulation method, the famous Gillespie's stochastic simulation algorithm (SSA). We proposed a Gillespie-type algorithm to simulate this kind of model. The resulting algorithm is supposed to provide us with equivalent results and have at least similar or even better performance, especially for systems with reaction rates span in different magnitudes. We generated simulation code for two different models: the May-Leonard model, which considers a four-state cyclically competitive game, and the Ziff-Gulari-Barshad (ZGB) model, which is used to study the catalytic surface-reaction process of CO and O_2. We compared the results obtained from simulation programmed using different algorithms, both phenomenologically and quantitatively, and concluded that these two different algorithms are roughly equivalent and provide us similar results. We also compared their performance using the CPU cost of codes im-

plemented using different methods. Our new proposed was less efficient than the original algorithm for the May-Leonard model case. But they are still comparable. However, it performed much better than the StochSim method when applied for the ZGB model, as long as the infinite reaction rate assumption is abandoned. We also analyze the reason behind this dramatically different performance fact.

The outlook of this topic lies mainly in two directions. Further modification can be made to improve the efficiency of this algorithm. For example, maintain the two-direction mapping structures between the particle-based space and reaction-based space is the main reason that our new proposed algorithm is inefficient, if one or both of these mapping can be simplified, its performance will be highly improved. The second direction is to consider the further application of this method and/or to check current existing results or conclusions in these stochastic lattice model studies that originally obtained using the StochSim algorithm. We believe our new method will open another possible way for research in this regime and contribute to future studies.

Bibliography

[1] A. M. Turing, Proceedings of the London mathematical society **2**, 230 (1937).

[2] M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods, Volume I* (John Wiley & Sons, 1986).

[3] B. D. Ripley, *Stochastic simulation*, Vol. 316 (John Wiley & Sons, 1987).

[4] S. S. Sawilowsky and G. Fahoome, *Statistics through Monte Carlo simulation with fortran* (2003).

[5] C. J. Morton-Firth and D. Bray, Journal of Theoretical Biology **192**, 117 (1998).

[6] H. H. McAdams and A. Arkin, Proceedings of the National Academy of Sciences **94**, 814 (1997).

[7] A. Arkin, J. Ross, and H. H. McAdams, Genetics **149**, 1633 (1998).

[8] N. Fedoroff and W. Fontana, Science **297**, 1129 (2002).

[9] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain, Science **297**, 1183 (2002).

[10] D. T. Gillespie, The Journal of Chemical Physics **115**, 1716 (2001).

[11] A. B. Stundzia and C. J. Lumsden, Journal of computational physics **127**, 196 (1996).

[12] D. A. McQuarrie, Journal of applied probability **4**, 413 (1967).

[13] D. T. Gillespie, Journal of computational physics **22**, 403 (1976).

[14] D. T. Gillespie, The journal of physical chemistry **81**, 2340 (1977).

[15] D. T. Gillespie, *Markov processes: an introduction for physical scientists* (Elsevier, 1991).

[16] M. A. Gibson and J. Bruck, The journal of physical chemistry A **104**, 1876 (2000).

[17] Y. Cao, H. Li, and L. Petzold, The journal of chemical physics **121**, 4059 (2004).

[18] D. T. Gillespie, Annu. Rev. Phys. Chem. **58**, 35 (2007).

[19] J. M. McCollum, G. D. Peterson, C. D. Cox, M. L. Simpson, and N. F. Samatova, Computational biology and chemistry **30**, 39 (2006).

[20] D. F. Anderson, The Journal of chemical physics **127**, 214107 (2007).

[21] A. Slepoy, A. P. Thompson, and S. J. Plimpton, The journal of chemical physics **128**, 05B618 (2008).

[22] D. T. Gillespie, A. Hellander, and L. R. Petzold, The Journal of chemical physics **138**, 05B201_1 (2013).

[23] Y. Cao, D. T. Gillespie, and L. R. Petzold, The Journal of chemical physics **122**, 014116 (2005).

[24] Y. Cao, D. T. Gillespie, and L. R. Petzold, The Journal of chemical physics **123**, 144917 (2005).

[25] H. Kuwahara and I. Mura, The Journal of chemical physics **129**, 10B619 (2008).

[26] D. T. Gillespie, M. Roh, and L. R. Petzold, The Journal of chemical physics **130**, 174103 (2009).

[27] M. K. Roh, D. T. Gillespie, and L. R. Petzold, The Journal of chemical physics **133**, 174106 (2010).

[28] C. V. Rao and A. P. Arkin, The Journal of chemical physics **118**, 4999 (2003).

[29] D. T. Gillespie and L. R. Petzold, The Journal of Chemical Physics **119**, 8229 (2003).

[30] Y. Cao, D. T. Gillespie, and L. R. Petzold, The Journal of chemical physics **124**, 044109 (2006).

[31] Y. Cao, D. T. Gillespie, and L. R. Petzold, The Journal of chemical physics **123**, 054104 (2005).

[32] B. Bayati, P. Chatelain, and P. Koumoutsakos, Journal of Computational Physics **228**, 5908 (2009).

[33] A. Auger, P. Chatelain, and P. Koumoutsakos, The Journal of chemical physics **125**, 084103 (2006).

[34] E. Mjolsness, D. Orendorff, P. Chatelain, and P. Koumoutsakos, The Journal of chemical physics **130**, 144110 (2009).

[35] D. F. Anderson, The Journal of chemical physics **128**, 054103 (2008).

[36] M. Rathinam, L. R. Petzold, Y. Cao, and D. T. Gillespie, The Journal of Chemical Physics **119**, 12784 (2003).

[37] Y. Cao, L. R. Petzold, M. Rathinam, and D. T. Gillespie, The Journal of chemical physics **121**, 12169 (2004).

[38] M. Rathinam, L. R. Petzold, Y. Cao, and D. T. Gillespie, Multiscale Modeling & Simulation **4**, 867 (2005).

[39] Y. Cao, D. T. Gillespie, and L. R. Petzold, The Journal of chemical physics **126**, 224101 (2007).

[40] C. J. Morton-Firth, T. S. Shimizu, and D. Bray, Journal of molecular biology **286**, 1059 (1999).

[41] N. Le Novere and T. S. Shimizu, Bioinformatics **17**, 575 (2001).

[42] D. P. Tolle and N. Le Novère, Current Bioinformatics **1**, 315 (2006).

[43] A. Chatterjee and D. G. Vlachos, Journal of computer-aided materials design **14**, 253 (2007).

[44] T. S. Shimizu and D. Bray, Foundations of Systems Biology , 213 (2001).

[45] M. F. Pettigrew and H. Resat, The Journal of chemical physics **123**, 114707 (2005).

[46] Z. Liu and Y. Cao, IET systems biology **2**, 334 (2008).

[47] C. Gardiner, K. McNeil, D. Walls, and I. Matheson, Journal of Statistical Physics **14**, 307 (1976).

[48] S. A. Isaacson and C. S. Peskin, SIAM Journal on Scientific Computing **28**, 47 (2006).

[49] S. Isaacson, D. McQueen, and C. S. Peskin, Proceedings of the National Academy of Sciences (2011).

[50] S. Engblom, L. Ferm, A. Hellander, and P. Lötstedt, SIAM Journal on Scientific Computing **31**, 1774 (2009).

[51] D. Bernstein, Physical Review E **71**, 041103 (2005).

[52] J. Elf, A. Doncic, and M. Ehrenberg, in *Fluctuations and noise in biological, biophysical, and biomedical systems*, Vol. 5110 (International Society for Optics and Photonics, 2003) pp. 114–125.

[53] R. M. May and W. J. Leonard, SIAM journal on applied mathematics **29**, 243 (1975).

[54] T. Reichenbach, M. Mobilia, and E. Frey, Nature **448**, 1046 (2007).

[55] T. Reichenbach, M. Mobilia, and E. Frey, Journal of Theoretical Biology **254**, 368 (2008).

[56] T. Reichenbach, M. Mobilia, and E. Frey, Physical review letters **99**, 238105 (2007).

[57] Q. He, M. Mobilia, and U. C. Täuber, The European Physical Journal B **82**, 97 (2011).

[58] S. Esmaeili, B. L. Brown, and M. Pleimling, Physical Review E **98**, 062105 (2018).

[59] A. Zaikin and A. Zhabotinsky, Nature **225**, 535 (1970).

[60] S. R. Serrao and U. C. Täuber, Journal of Physics A: Mathematical and Theoretical **50**, 404005 (2017).

[61] H. Chaté and P. Manneville, Physica A: Statistical Mechanics and its Applications **224**, 348 (1996).

[62] M. Mobilia, A. M. Rucklidge, and B. Szczesny, Games **7**, 24 (2016).

[63] B. Sinervo, D. B. Miles, W. A. Frankino, M. Klukowski, and D. F. DeNardo, Hormones and Behavior **38**, 222 (2000).

[64] D. M. Wolf, V. V. Vazirani, and A. P. Arkin, Journal of theoretical biology **234**, 227 (2005).

[65] C. N. Satterfield, *Heterogeneous catalysis in practice* (McGraw-Hill Companies, 1980).

[66] R. M. Ziff, E. Gulari, and Y. Barshad, Physical Review Letters **56**, 2553 (1986).