

A NUMERICAL STUDY OF THE STOCHASTIC REACTION-DIFFUSION MASTER
EQUATION USING TENSORS AND PARALLELISM WITH APPLICATION TO
BIOLOGICAL MODELS

by

MD MUSTAFIJUR RAHMAN

ROGER B. SIDJE, COMMITTEE CHAIR

MIN SUN

DAVID HALPERN

MOJDEH RASOULZADEH

MILOUD SADKANE

A DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Mathematics
in the Graduate School of
The University of Alabama

TUSCALOOSA, ALABAMA

2025

Copyright Md Mustafijur Rahman 2025
ALL RIGHTS RESERVED

ABSTRACT

Biological systems often exhibit both stochasticity in chemical reactions and spatial diffusion of molecules, making their modeling inherently complex. The Reaction-Diffusion Master Equation (RDME) provides a stochastic framework to describe such systems by partitioning space into compartments and considering well-mixed species within each. Compared to the Chemical Master Equation (CME), the RDME introduces a significantly larger state space due to the inclusion of jump processes between compartments. The Stochastic Simulation Algorithm (SSA) is commonly used for CME analysis, but is computationally intensive for large-scale systems, which grow worse in RDME problems. Given this complexity, efficient numerical methods are crucial for analyzing and predicting the dynamics of the systems. In this dissertation, we explore biological models using the RDME formulation and leverage tensor-based techniques to manage the large state space efficiently. Tensors, as multidimensional arrays, offer powerful mechanisms for processing and analyzing data in complex biological systems, making them well-suited for RDME applications. To address the computational challenges associated with simulating RDME trajectories over time with the SSA, we implement a parallelized approach using OpenMP with FORTRAN, distributing computations across multiple processors to enhance efficiency. Building on tensor-based methods and high-performance computing, our study aims to significantly accelerate RDME simulations while maintaining accuracy. Advanced numerical techniques enable more effective modeling of stochastic reaction-diffusion systems, providing deeper insights into the dynamics of biological processes.

DEDICATION

To my parents, brother, grandpa, relatives, friends, and mentors.

ACKNOWLEDGMENTS

I am deeply grateful to my advisor, Dr. Roger B. Sidje, for his unwavering encouragement throughout this challenging yet often rewarding journey. His guidance has been invaluable in helping me appreciate the importance of attention to detail, develop the craft of scientific writing, and learn numerous essential skills that I continue to refine.

I also extend my sincere appreciation to my committee members for their time and support. I am thankful to Dr. Miloud Sadkane from the Université de Brest (France) for serving as the external member, as well as to Dr. David Halpern, Dr. Min Sun, and Dr. Mojdeh Rasoulzadeh from the Department of Mathematics at the University of Alabama for their invaluable guidance and contributions.

I would also like to express my heartfelt gratitude to all the teachers who have guided me from my earliest years through higher education. Their dedication, encouragement, and belief in my potential laid the foundation for everything I have achieved. I am equally thankful to my extended family and friends, whose kindness, support, and uplifting words have sustained me through both challenges and triumphs. Their presence—whether near or far—has been a steady source of motivation and comfort throughout this journey.

Lastly, I am deeply thankful to my wonderful and loving family. My father, mother, and brother have always supported me and believed in my endeavors. Their unwavering encouragement and faith in me have been a source of strength, especially during these past years in graduate school. They have consistently emphasized positivity and motivated me to pursue my dreams.

CONTENTS

ABSTRACT	ii
DEDICATION	iii
ACKNOWLEDGMENTS	iv
LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER 1 Introduction	1
CHAPTER 2 Chemical Reaction Systems Incorporating Diffusion as a Jump Process	5
2.1 Chemical Reactions	5
2.2 Types of Reactions	5
2.2.1 Isomerization	6
2.2.2 Dimerization	6
2.2.3 Synthesis	6
2.2.4 Decomposition	6
2.2.5 Degradation	7
2.2.6 Production	7
2.3 Randomness in Chemical Reaction	7
2.4 Stochastic Simulation of Degradation	8
2.5 Stochastic Simulation of Production and Degradation	9
2.6 Diffusion	11
2.7 Compartment-Based Diffusion	12

2.8	Chemical Master Equation	14
2.9	Michaelis-Menten Enzyme Kinetics	17
2.10	Reaction-Diffusion Master Equation	20
2.11	Birth-Death Process	21
2.12	Spatiotemporal Chemical Master Equation	24
2.12.1	Metastable Compartments	24
2.12.2	Particle Based Reaction-Diffusion	24
2.12.3	ST-CME	24
2.12.4	Binding and Unbinding	25
CHAPTER 3 A Study of a Metapopulation Model Using the Stochastic Reaction-Diffusion Master Equation		29
3.1	Stochastic Simulation Algorithm of Gillespie	29
3.2	Finite State Projection	30
3.3	Metapopulation Model	32
3.4	Integrating RDME into Metapopulation Model	33
3.5	Numerical Results	36
3.6	Discussion and Conclusion	37
CHAPTER 4 An Application of Tensors in the Stochastic Reaction-Diffusion Master Equation		39
4.1	Matrix Products	39
4.1.1	Hadamard Product	39
4.1.2	Kronecker Product	40
4.1.3	Khatri-Rao Product	40
4.2	Tensor Notations and Definitions	40
4.2.1	Tensor	40
4.2.2	Subtensor	41

4.2.3	Fiber	41
4.2.4	Slice	42
4.2.5	n -rank of a Tensor	43
4.2.6	The reshape and vec Operations on Tensors	43
4.3	Tensor Decomposition	44
4.3.1	Higher Order Singular Value Decomposition (HOSVD)	45
4.3.2	CANDECOMP/PARAFAC (CP) Decomposition	46
4.3.3	Tensor Train (TT) Decomposition	47
4.4	Transient Matrix in Kronecker Product Form	47
4.5	Study Findings using a Metapopulation Model	49
4.6	Discussion and Conclusion	52
CHAPTER 5 Parallel Implementation of the Stochastic Reaction-Diffusion Master Equation		55
5.1	History of Parallel Computing	55
5.2	What is Parallel Computing	56
5.3	Types of Parallel Computing	57
5.4	OpenMP	58
5.5	Alabama Supercomputer Center (ASC)	62
5.5.1	About ASC	62
5.5.2	Accessing the ASC	63
5.5.3	Necessary Linux Commands	64
5.5.4	Submitting Job into Queue	65
5.6	Parallel Stochastic Simulation Algorithm	66
5.7	Speed-Up and Parallel Efficiency	67
5.8	Numerical Results	68

5.8.1	Metapopulation Model	68
5.8.2	Birth-Death Process	70
5.8.3	Schögl Model	73
5.8.4	SIR Model	78
5.9	Discussion and Conclusion	82
CHAPTER 6 CONCLUSION		83
REFERENCES		85

LIST OF TABLES

2.1	Reactions and propensities of Michaelis-Menten model.	18
2.2	Reactions, propensities and state change vectors of birth-death model.	23
3.1	Reactions, propensities and state change vectors of metapopulation model.	34
5.1	Computational time with different number of cores for parallel stochastic simulation of metapopulation model.	68
5.2	Computational time with different number of cores for parallel stochastic simulation of birth-death model.	71
5.3	Reactions, propensities and state change vectors of Schögl model.	74
5.4	Computational time with different number of cores for parallel stochastic simulation of Schögl model.	76
5.5	Reactions, propensities, and state change vectors of SIR model.	79
5.6	Computational time with different number of cores for parallel stochastic simulation of SIR model.	80

LIST OF FIGURES

2.1	Three realizations of SSA for degradation.	9
2.2	Five realizations of SSA for production and degradation.	11
2.3	Number of molecules in 40 compartments using the compartment-based SSA.	14
2.4	Transitions to all possible states of Michaelis-Menten model with initial state $[2, 1, 0, 0]$	18
2.5	Probabilities of all possible states of Michaelis-Menten model with $t = 10$	20
2.6	Illustration of RDME in birth-death model.	22
2.7	Marginal probabilities of the molecules of birth-death model after $t = 50$	23
2.8	Eukaryotic cell (image from genome.gov).	26
2.9	Evolution of species A (image adapted from [74]).	27
2.10	Evolution of species B (image adapted from [74]).	27
2.11	Evolution of species C (image adapted from [74]).	28
3.1	Illustration of RDME in metapopulation model.	33
3.2	Visualization of the sparsity pattern of the transition matrix in RDME.	35
3.3	Marginal probability distribution of the number of healthy and infected individuals in two areas using FSP at time $t = 10$	36
3.4	Marginal probability distribution of the number of healthy and infected individuals in two areas using SSA at time $t = 10$	37
4.1	Visualization of different tensors (image from packtpub.com).	41
4.2	Different fibers of a 3-mode tensor.	42
4.3	Different slices of a 3-mode tensor.	43

4.4	Marginal probability distribution of the number of healthy and infected individuals in two areas of a metapopulation model using FSP at time $t = 10$.	51
4.5	Marginal probability distribution of the number of healthy and infected individuals in two areas of a metapopulation model using tensors at time $t = 10$.	52
4.6	Comparison of marginal probability distribution of the number of healthy and infected individuals in two areas of a metapopulation model using FSP and tensors.	53
5.1	Parallel demonstration.	60
5.2	Nested parallel demonstration.	61
5.3	Parallel do demonstration.	62
5.4	Accessing the ASC using terminal window.	63
5.5	Two step authentication before login.	63
5.6	Information upon successful login.	64
5.7	Selecting desired parallel environment for the job.	65
5.8	Submitted job details.	66
5.9	Decreasing computational time with different number of cores for parallel stochastic simulation of metapopulation model.	69
5.10	Computational speed-up of parallel implementation in metapopulation model.	69
5.11	Parallel efficiency in metapopulation model.	70
5.12	Decreasing computational time with different number of cores for parallel stochastic simulation of birth-death model.	71
5.13	Computational speed-up of parallel implementation in birth-death model.	72
5.14	Parallel efficiency in birth-death model.	72
5.15	Decreasing computational time with different number of cores for parallel stochastic simulation of Schögl model.	76
5.16	Computational speed-up of parallel implementation in Schögl model.	77
5.17	Parallel efficiency in Schögl model.	77

5.18	Decreasing computational time with different number of cores for parallel stochastic simulation of SIR model.	80
5.19	Computational speed-up of parallel implementation in SIR model.	81
5.20	Parallel efficiency in SIR model.	81

CHAPTER 1 INTRODUCTION

Biological systems at the cellular level involve random interactions among species that have small copy numbers. This fact has been well-established in the area of molecular cell biology through both experimental investigations and mathematical modeling. The spatial environment within biological cells is highly intricate and consists of geometrically complex structures. The chemical master equation (CME) [33, 48, 75] is a mathematical framework used to describe the stochastic behavior of chemical reactions in a well-mixed system. It provides a way to model the time evolution of the probability distribution of the number of molecules of each species in a system, taking into account both the discrete nature of molecules and the stochasticity of their interactions.

In recent years, there has been considerable attention given to the development of effective computational techniques for performing discrete stochastic simulation of well-mixed biochemical systems [34]. The stochastic simulation algorithm (SSA) [25, 31, 32] is a computational method employed to obtain sample realizations guided by the chemical kinetics that underpin the CME. The SSA is a powerful tool for simulating complex biochemical networks, as it can take into account the stochasticity of the system and can generate statistically meaningful results. We refer to [43] for a demonstration of an implementation to the Michaelis-Menten system. Despite being useful, the SSA algorithm is time consuming since it is based on realizations of the states over time. Acceleration techniques such as tau-leaping have been introduced to make this algorithm faster [11, 54]. The finite state projection algorithm (FSP) [20, 27, 37, 57] is another way of tackling the CME where it involves solving a set of differential equations that describe the change in the probability distribution over time. The equations are based on the transition rates

between states in the system and the current probability distribution. The algorithm can be used to compute the probability of reaching a specific state or a set of states within a given time interval.

Unlike traditional chemical reaction models that assume that reactions occur instantaneously and uniformly throughout the system, cellular environments are not uniform in space, and spatial heterogeneity is a common feature. In numerous biochemical systems, the spatial heterogeneity of species cannot be disregarded, rendering the systems not well-stirred. This may be due to the slow transport of molecules through the solvent compared to typical reaction times or the strong localization of some reactions. Experimental data from various sources [22, 23, 56], demonstrate the significance of considering both the stochastic characteristics and spatial distribution of a system to explain its behavior. Several mathematical methods, which are different but interrelated [3, 24, 69], have recently been employed to describe stochastic reaction-diffusion systems in biological cells [61, 68]. Here, the assumption is that molecules in stochastic reaction-diffusion systems behave as points that undergo continuous Brownian motion in space. Whenever two molecules come within a defined reaction radius, bimolecular chemical reactions occur instantly. This modeling approach is referred to as spatially continuous diffusion-limited reaction (SCDLR), which was originally proposed by Smoluchowski [66]. In contrast to these methods, when we treat the diffusion at a molecular level as a special set of reactions in the CME, we arrive at the reaction-diffusion master equation (RDME) [5, 28, 29, 41, 42, 46, 47, 64, 65, 74].

RDME is a mathematical framework used to describe the dynamics of stochastic chemical reaction systems that also involve diffusion processes. The equation describes the time-dependent probability distribution function (PDF) of the system state, similar to the CME, but with a significantly higher dimensionality. In RDME, the space is divided into compartments. Each compartment is considered to be well-mixed and reactions within a particular compartment are considered to be consistent with the homogeneous case. Also,

molecules can get diffusive transfers between adjacent compartments. The state of the system at any given time is defined by the number of molecules of each species in each compartment. Mathematically, the RDME is the forward Kolmogorov equation for a continuous-time jump Markov process. To tackle the RDME one feasible way is to generate samples from the RDME and collect statistics in a Monte Carlo fashion. The works in [8, 67] illustrate some examples of the SSA applied to reaction-diffusion systems in one space dimension. The next reaction method (NRM) [30] is a version of the SSA that is known for its efficiency. A software called MesoRD [39] offers a specialized implementation of NRM, which is designed for diffusive systems and known as the next sub-volume method (NSM) [24]. The aim of the binomial tau-leap spatial stochastic simulation algorithm [55] is to enhance its performance by merging the concepts of consolidating diffusive transfers and utilizing the priority queue structure from the NSM. In certain situations, it may be feasible to consider diffusion in a deterministic manner, thereby minimizing the need to monitor fast diffusive transfers to a great extent. The reactions are handled by SSA in those situations. This approach is exemplified by the hybrid multi-scale kinetic Monte Carlo method [76] and the Gillespie multi-particle method [63]. The corresponding macroscopic equation for the concentrations of the species is the reaction-diffusion equation (RDE) which is a partial differential equation (PDE). This equation is employed in diverse applications and is valid for large numbers of molecules when stochastic effects can be neglected.

Below is the outline of the dissertation,

- Chapter 2 incorporates the different types of chemical reactions happen in biological models. The derivation of CME is shown with a case study. In addition, when diffusion is added as a jump process, that leads to the RDME.
- Chapter 3 is devoted to a study of a particular Metapopulation model in which the RDME framework has been integrated. Numerical techniques like the SSA and FSP is used to analyze the marginal probability distributions.

- Chapter 4 details the tensor terminologies and different operations with tensors. Application of tensors in RDME is shown using a biological model.
- Chapter 5 provides the concepts of different types of parallel computing. The parallel SSA algorithm is employed using supercomputer to accelerate numerical simulation run-times.
- Finally, Chapter 6 summarizes the findings of the dissertation and provides potential for future direction.

CHAPTER 2 CHEMICAL REACTION SYSTEMS INCORPORATING DIFFUSION AS A JUMP PROCESS

2.1 Chemical Reactions

Chemical reactions are essential to a wide range of natural and biological processes, taking place at both macroscopic and microscopic levels. These reactions entail the conversion of substances via the reorganization of atomic or molecular structures, resulting in the creation of novel chemical species. In a system where molecules are uniformly distributed, reaction dynamics are often described using deterministic models based on the law of mass action, which assumes that reaction rates are proportional to the concentrations of reactants. However, such models become less accurate in situations where molecular interactions are influenced by spatial constraints or when the number of reacting molecules is small enough that random fluctuations play a significant role.

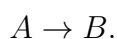
In real-world systems, chemical reactions rarely occur in isolation. Instead, they engage inside intricate networks where several interactions occur concurrently, influencing one another in ways that can lead to emergent behaviors such as oscillatory dynamics, feedback loops, or self-organizing patterns. These interrelated reactions are fundamental to numerous biological and environmental systems, necessitating mathematical models that incorporate reaction kinetics alongside with stochastic fluctuations.

2.2 Types of Reactions

Reactions can proceed in a single direction, leading to a complete transformation of reactants into products, or they can be reversible, meaning that the products can recombine to regenerate the original reactants. Some chemical reactions involve only a

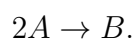
single species, while others require interactions between multiple distinct species. Consider A , B , C be three different species. The arrow stands to indicate a reaction where reactants are placed on the left-hand side and the products appear on the right-hand side. Below are some of the most common types of reactions.

2.2.1 Isomerization



In isomerization, a molecule changes its structural arrangement without altering its molecular formula and transform into another molecule. Isomerization plays a crucial role in biological pathways, such as glucose metabolism.

2.2.2 Dimerization



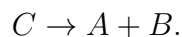
When two identical molecules chemically bond to form a dimer then the process is called dimerization. Dimerization can occur spontaneously or be facilitated by external factors such as catalysts, pressure, or temperature.

2.2.3 Synthesis



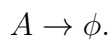
A reaction where two different molecules combine to form a more complex product. This type of reaction is essential in organic chemistry, material science, and biological processes.

2.2.4 Decomposition



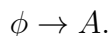
In decomposition, a complex compound breaks down into different simpler substances. This reaction is the opposite of Synthesis.

2.2.5 Degradation



This represents the removal or disappearance of a species from a system. This can occur through decay or consumption by another reactant. Enzymes often facilitate degradation in biological systems, while chemical or physical agents drive it in industrial applications.

2.2.6 Production



Production is a process where new molecule is introduced into a system. This is also called as birth reaction. In stochastic models, birth reactions are often used to describe population growth, where new individuals appear at a given rate.

2.3 Randomness in Chemical Reaction

Continuous modeling of a chemical reaction system allows us to depict the concentration of a species across time. Real chemical reactions, on the other hand, have discrete molecule interactions whereby populations vary by entire integer quantities instead of fractions. Deterministic modeling explains, from present and past conditions, how a system develops. Under this method, preceding conditions cause changes in molecular populations that produce smooth graphical depictions of species concentrations across time. This approach, however, presupposes constant changes and might result in erroneous interpretations such modeling fractional molecules, so it does not entirely reflect the reality of chemical reactions. Stochastic modeling, on the other hand, acknowledges randomness and the

impossibility of constantly exactly predicting future states. Here the next state of the system depends just on its current state, generating varying graphical representations that reflect the natural unpredictability seen in actual chemical reactions.

Randomness plays a crucial role in chemical reactions, as molecular interactions are inherently probabilistic. At microscopic scales, reactions occur due to random collisions between molecules, influenced by factors such as temperature, concentration, and diffusion. This unpredictability makes stochastic modeling essential for accurately capturing reaction dynamics, especially in systems with low molecular counts where fluctuations significantly impact behavior. Unlike deterministic models, which assume smooth concentration changes, stochastic approaches account for the discrete and probabilistic nature of chemical processes, leading to more realistic predictions. Embracing randomness allows for a deeper understanding of reaction mechanisms, variability in biological systems, and the emergence of complex behaviors in nature.

2.4 Stochastic Simulation of Degradation

Let us consider the single chemical reaction,



where c is the rate constant of the reaction and S is the chemical species that is degrading. we denote the number of molecules of S at time t as $S(t)$. dt is an infinitesimally small time step and $c dt$ gives the probability that a randomly chosen molecule of S degraded during the time interval $[t, t + dt)$. In particular, The probability that exactly one reaction (2.1) occurs during the time interval $[t, t + dt)$ is equal to $S(t)c dt$. Our aim is to compute the number of molecules $S(t)$ for times $t > 0$. For that purpose we need to generate random numbers uniformly distributed in the interval $(0, 1)$. We choose a small time step Δt and using the following two steps of stochastic simulation,

we compute the number of molecules $S(t)$ at times $t = i\Delta t, i = 1, 2, 3, \dots$

1. Generate a random number r uniformly distributed in the interval $(0, 1)$.
2. If $r < S(t)c\Delta t$, then put $S(t + \Delta t) = S(t) - 1$; otherwise, $S(t + \Delta t) = S(t)$. Then continue with step 1 for time $t + \Delta t$.

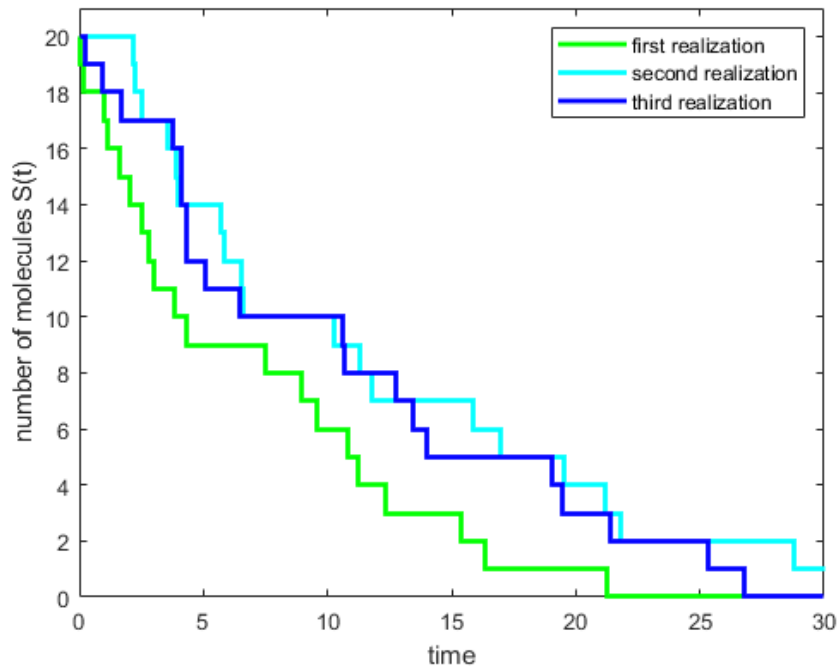


Figure 2.1: Three realizations of SSA for degradation.

Choosing $c = 0.1$, $S(0) = 20$ and $\Delta t = 0.005$, we repeat the stochastic simulation three times and we plot three realizations of SSA. In the figure 2.1, we notice that the realizations are different each time we run the algorithm and that is obvious for any SSA.

2.5 Stochastic Simulation of Production and Degradation

Consider a system of two chemical reactions



The first reaction in (2.2) describes the degradation of chemical S with the rate constant c_1 and the second reaction represents the creation of chemical S with the rate constant c_2 . Here $c_1 dt$ gives the probability that a randomly chosen molecule of S degraded during the time interval $[t, t + dt)$ and $c_2 dt$ gives the probability that a randomly chosen molecule of S produced during the time interval $[t, t + dt)$. Since we have two reactions here, our goal is to find what reaction will occur next as well as the time. We find τ such that $t + \tau$ is the time when the next reaction occurs given that $S(t)$ is the molecules present at time t . The stochastic simulation steps for this reaction scheme are,

1. Generate two random numbers r_1, r_2 uniformly distributed in $(0, 1)$.
2. Compute $a_0 = S(t)c_1 + c_2$.
3. Compute the time when the next chemical reaction takes place as $t + \tau$ where

$$\tau = \frac{1}{a_0} \ln \left[\frac{1}{r_1} \right]. \quad (2.3)$$

4. Compute the number of molecules at time $t + \tau$ by

$$S(t + \tau) = \begin{cases} S(t) + 1, & \text{if } r_2 < c_2/a_0 \\ S(t) - 1, & \text{if } r_2 \geq c_2/a_0 \end{cases}$$

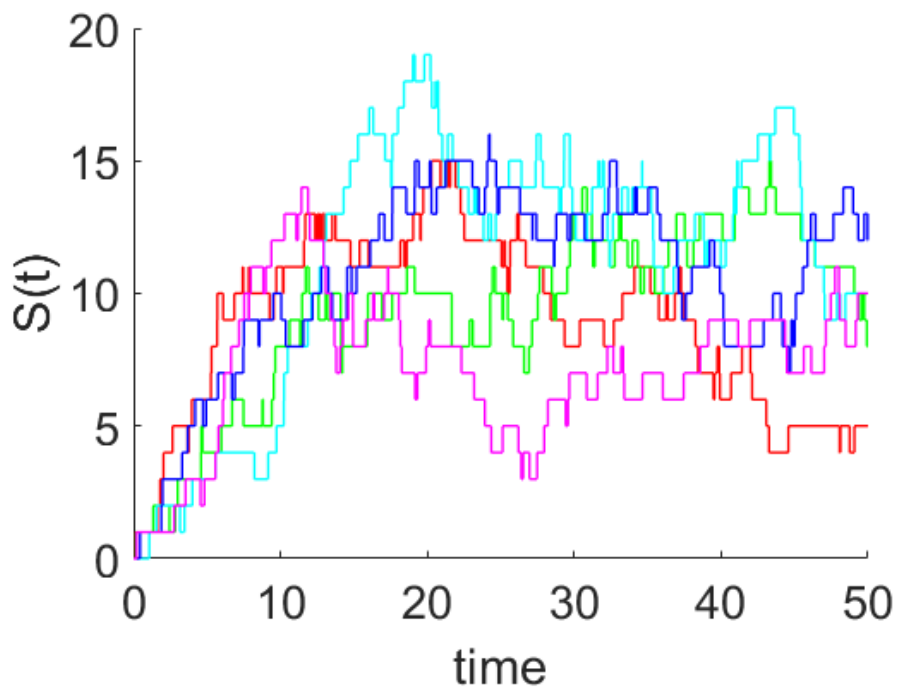


Figure 2.2: Five realizations of SSA for production and degradation.

We present five realizations of SSA in figure 2.2 for $S(0) = 0$, $c_1 = 0.1$ and $c_2 = 1$. We see that, after an initial transient, the number of molecules $S(t)$ are following a trend which is actually the fluctuation around the mean value.

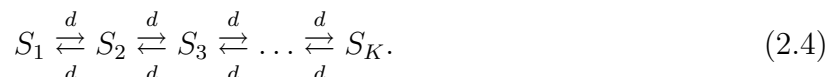
2.6 Diffusion

Diffusion is the process whereby particles, molecules, or other entities disperse from an area of higher concentration to an area of lower concentration. It is the random migration of molecules arising from motion due to thermal energy [7]. This is a key transport process found in diverse physical, chemical, and biological systems. Diffusion is mathematically characterized by Fick's equations, which connect particle flux to the concentration gradient, or by stochastic processes like Brownian motion. In continuous systems, diffusion is regulated by the diffusion equation, a partial differential equation that describes the temporal dispersion of a substance. In discrete or stochastic contexts, diffusion can be represented as a jump process, wherein particles shift between discrete states with specified

probabilities. Diffusion is essential in various natural and artificial processes, such as gas exchange in biological systems, heat transmission, chemical reactions, and the distribution of compounds in fluids and porous media.

2.7 Compartment-Based Diffusion

Consider the following chemical reactions,



where

$$S_i \xrightleftharpoons[d]{d} S_{i+1} \quad \text{means that} \quad S_i \xrightarrow{d} S_{i+1} \quad \text{and} \quad S_{i+1} \xrightarrow{d} S_i.$$

We see the reactions (2.4) are diffusive jumps. Consider the computational domain $[0, L]$. We choose $L = 1$ and split the domain into $K = 40$ compartments of length $h = L/K = 1/40 = .025$. We label the number of molecules of chemical species S in the i -th compartment $[(i-1)h, ih)$ by $S_i, i = 1, \dots, K$. The compartment-based SSA has six steps.

1. Generate two random numbers r_1, r_2 uniformly distributed in $(0, 1)$.
2. Compute propensity functions of reactions by $a_i = S_i(t)d$ for $i = 1, 2, \dots, K$.

Compute

$$a_0 = \sum_{i=1}^{K-1} a_i + \sum_{i=2}^K a_i.$$

3. Compute the time when the next chemical reaction occurs as $t + \tau$ where τ is given by equation (2.3).
4. If $r_2 < \sum_{i=1}^{K-1} a_i/a_0$, then find $j \in \{1, 2, \dots, K-1\}$ such that

$$r_2 \geq \frac{1}{a_0} \sum_{i=1}^{j-1} a_i \quad \text{and} \quad r_2 < \frac{1}{a_0} \sum_{i=1}^j a_i.$$

Then compute the number of molecules at time $t + \tau$ by

$$\begin{aligned} S_j(t + \tau) &= S_j(t) - 1, \\ S_{j+1}(t + \tau) &= S_{j+1}(t) + 1, \\ S_i(t + \tau) &= S_i(t), \quad \text{for } i \neq j, i \neq j + 1. \end{aligned}$$

5. If $r_2 \geq \sum_{i=1}^{K-1} a_i/a_0$, then find $j \in \{2, 3, \dots, K\}$ such that

$$r_2 \geq \frac{1}{a_0} \left(\sum_{i=1}^{K-1} a_i + \sum_{i=2}^{j-1} a_i \right) \quad \text{and} \quad r_2 < \frac{1}{a_0} \left(\sum_{i=1}^{K-1} a_i + \sum_{i=2}^j a_i \right).$$

Then compute the number of molecules at time $t + \tau$ by

$$\begin{aligned} S_j(t + \tau) &= S_j(t) - 1, \\ S_{j-1}(t + \tau) &= S_{j-1}(t) + 1, \\ S_i(t + \tau) &= S_i(t), \quad \text{for } i \neq j, i \neq j - 1. \end{aligned}$$

6. Continue with step 1 for time $t + \tau$.

We choose $d = 0.16$ and consider initially there are 150 molecules in compartment 16, 150 molecules in compartment 17, and no molecules present in other compartments. Figure 2.3 shows the number of molecules in each compartment at time $t = 4$ by using the compartment-based SSA.

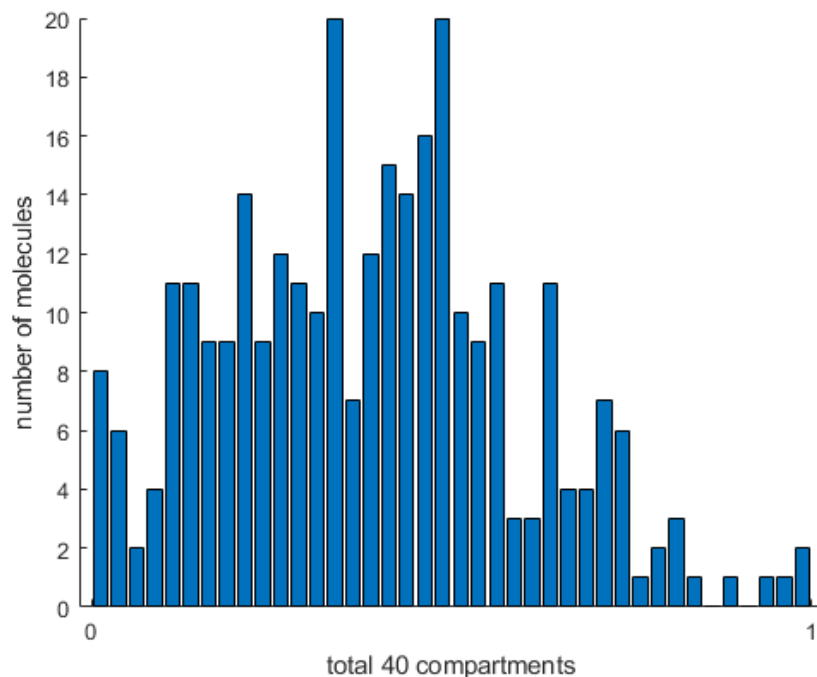


Figure 2.3: Number of molecules in 40 compartments using the compartment-based SSA.

2.8 Chemical Master Equation

The chemical master equation (CME) [33] is a fundamental framework for modeling the stochastic behavior of chemical reaction systems at a molecular level. It describes how the probability distribution of different molecular populations evolves over time, accounting for the inherent randomness in reaction events. Unlike deterministic approaches, which rely on continuous concentration changes, the CME models reactions as discrete and probabilistic transitions between different molecular states. This makes it particularly useful for systems with small molecule counts, where fluctuations play a crucial role in system behavior.

Consider a system involving N molecular species $\{S_1, \dots, S_N\}$, represented by the state vector $\mathbf{X}(t) = [X_1(t), \dots, X_N(t)]^T$, where each $X_i(t)$ is a non-negative integer denoting the number of molecules of species S_i at time t . The state space of the system is the set of distinct vectors that correspond to the possible configurations the system can have. As many types of reactions occur, the system evolves from state to state through an

underlying Markov chain. Suppose there are M reaction channels, denoted by $\{R_1, \dots, R_M\}$, and assume that the system is well mixed and in thermal equilibrium. The dynamics of the reaction channel R_j is characterized by two factors: the propensity function a_j , which describes the likelihood of the reaction occurring given the current state of the system, and the state change (stoichiometric) vector $\boldsymbol{\nu}_j = [\nu_{1j}, \dots, \nu_{Nj}]^T$, which specifies the change in the number of molecules of each species that results from one instance of the reaction; $a_j(\mathbf{x})dt$ gives the probability that, given $\mathbf{X}(t) = \mathbf{x}$, one R_j reaction will occur in the next infinitesimal time interval $[t, t + dt]$, and ν_{ij} gives the change in X_i induced by one R_j reaction. That means the components of the state change vector are integers displaying the increase or decrease of the number of copies of each species after the associated reaction occurs.

We can now investigate the quantity $P(\mathbf{x}, t)$, which we define to be the probability that $\mathbf{X}(t) = \mathbf{x}$. Consider that the initial state vector $\mathbf{X}(0)$ is given. We aim to derive a recurrence. Assuming dt is sufficiently small that at most one reaction may occur over $[t, t + dt)$, we will work out the probability of being in state \mathbf{x} at time $t + dt$ knowing the likelihood of being in any of the possible states at time t . The first step is to realize that to be in state \mathbf{x} at time $t + dt$ there are only two scenarios for time t ; either the system was already in state \mathbf{x} at time t and no reaction happened over $[t, t + dt)$, or for some $1 \leq j \leq M$ the system was in state $\mathbf{x} - \boldsymbol{\nu}_j$ at time t and the j the reaction took place over $[t, t + dt)$, thereby bringing the system into state \mathbf{x} . We need to use the law of total probability, a classic result from probability theory. Suppose A is the event of interest and suppose that the events $E_0, E_1, E_2, \dots, E_M, E_{M+1}$ are disjoint (no more than one can happen) and exhaustive (one of them must happen). Then the law of total probability states that

$$P(A) = \sum_{j=0}^{M+1} P(A | E_j) P(E_j). \quad (2.5)$$

Here, $P(A | E_j)$ is such that the probability that A happens, given that E_j happens. In our case, A is the event that the system is in state \mathbf{x} at time $t + dt$. We assume E_0 be the

event that the system is in state \mathbf{x} at time t , let E_j for $1 \leq j \leq M$ be the event that the system is in state $\mathbf{x} - \boldsymbol{\nu}_j$ at time t , and let E_{M+1} be the event that the system is in any other state at time t . Now, for $1 \leq j \leq M$, $P(A | E_j)$ is simply the probability of the j reaction firing over $[t, t + dt)$. Drawing from the definition of the propensity functions this implies

$$P(A | E_j) = a_j(\mathbf{x} - \boldsymbol{\nu}_j) dt, \quad 1 \leq j \leq M. \quad (2.6)$$

Similarly, $P(A | E_0)$ is the probability of no reaction happening over $[t, t + dt)$. This must equal 1 minus the probability of any reaction firing, hence

$$P(A | E_0) = 1 - \sum_{j=1}^M a_j(\mathbf{x}) dt. \quad (2.7)$$

Consequently,

$$P(A | E_{M+1}) = 0, \quad (2.8)$$

because E_{M+1} contains all the states that are more than one reaction away from \mathbf{x} . Using (2.6), (2.7), and (2.8) in (2.5), along with the definition of $P(\mathbf{x}, t)$, we find that

$$P(\mathbf{x}, t + dt) = \left(1 - \sum_{j=1}^M a_j(\mathbf{x}) dt\right) P(\mathbf{x}, t) + \sum_{j=1}^M a_j(\mathbf{x} - \boldsymbol{\nu}_j) dt P(\mathbf{x} - \boldsymbol{\nu}_j, t). \quad (2.9)$$

This equation can be rearranged to

$$\frac{P(\mathbf{x}, t + dt) - P(\mathbf{x}, t)}{dt} = \sum_{j=1}^M (a_j(\mathbf{x} - \boldsymbol{\nu}_j) P(\mathbf{x} - \boldsymbol{\nu}_j, t) - a_j(\mathbf{x}) P(\mathbf{x}, t)). \quad (2.10)$$

Letting $dt \rightarrow 0$ we observe that the left-hand side of this equation becomes a time derivative, producing the CME

$$\frac{dP(\mathbf{x}, t)}{dt} = \sum_{j=1}^M [a_j(\mathbf{x} - \nu_j) P(\mathbf{x} - \nu_j, t) - a_j(\mathbf{x}) P(\mathbf{x}, t)]. \quad (2.11)$$

Equation (2.11) may be written in an equivalent matrix–vector form by enumerating all the states. If there are n possible states, $\mathbf{x}_1, \dots, \mathbf{x}_n$, the CME takes the form of a system of linear ordinary differential equations (ODEs),

$$\dot{\mathbf{P}}(t) = \mathcal{M} \cdot \mathbf{P}(t), \quad (2.12)$$

where $\mathcal{M} = [m_{ij}]$ is occupied by the propensities and represents the infinitesimal generator of the corresponding Markov chain. The probability vector $\mathbf{P} = [p_1, \dots, p_n]^T$ is such that each component $p_i = P(\mathbf{x}_i, t) = \text{Prob}(\mathbf{x}(t) = \mathbf{x}_i)$, the probability of being at state \mathbf{x}_i at time t , for $i = 1, \dots, n$. Given an initial $\mathbf{P}(0)$ the solution of (2.12) at time t is:

$$\mathbf{P}(t) = \exp(t\mathcal{M}) \mathbf{P}(0), \quad (2.13)$$

where the exponential of a bounded operator is defined by a Taylor series.

2.9 Michaelis-Menten Enzyme Kinetics

One of the classic example to demonstrate the utilization of CME is the Michaelis-Menten model. Maud Leonora Menten, one of the first women in Canada to earn a medical doctorate, later obtained a PhD in biochemistry and made significant contributions to pathology. She worked with eminent biologist and biographer Leonor Michaelis at the University of Berlin. Together, they produced a revolutionary study on enzyme kinetics in 1913 proving that, at a given concentration, every enzyme has a unique substrate and reaction rate. Now known as the Michaelis-Menten constant [15, 16], this idea set the groundwork for contemporary enzyme kinetics. But Henri and Brown [9, 10], who investigated enzyme activity in sugar combinations, initially devised the original

equation. Their mathematical approach made enzyme processes more comprehensively understandable. Before moving on to more complicated systems, stochastic simulations are studied from a basic model.

The Michaelis-Menten model has four species and three reactions. The species are defined as S = Substrate, E = Enzyme, ES = Enzyme-Substrate Complex, and P = Product. In this system, an enzyme converts all of its substrates into a product. The state of the system is the vector listing the population of each of the species, $\mathbf{x} = [S, E, ES, P]$, in this particular order.

Table 2.1: Reactions and propensities of Michaelis-Menten model.

Reaction	Propensity
$S + E \xrightarrow{c_1} ES$	$c_1 \times S \times E$
$ES \xrightarrow{c_2} S + E$	$c_2 \times ES$
$ES \xrightarrow{c_3} P + E$	$c_3 \times ES$

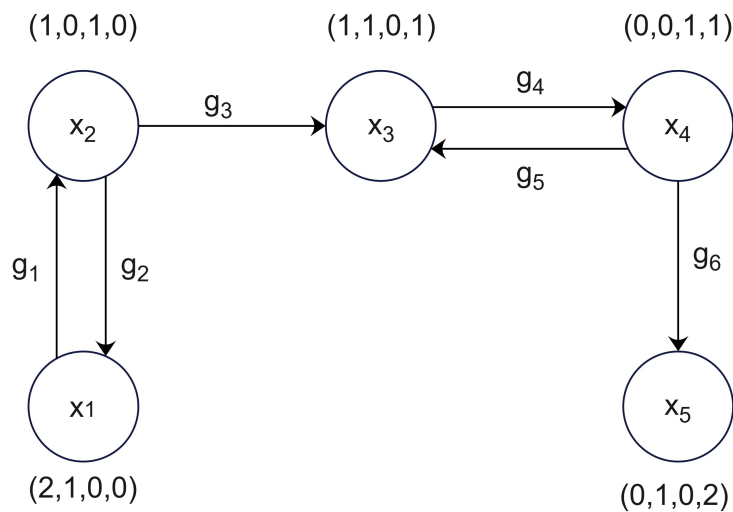


Figure 2.4: Transitions to all possible states of Michaelis-Menten model with initial state $[2, 1, 0, 0]$.

For experimental purpose we choose the initial state vector $[S, E, ES, P]$ to be $[2, 1, 0, 0]$.

Figure 2.4 tracks all the possible chemical reactions that can happen and their resulting states, \mathbf{x}_i , where $i = 1, \dots, 5$. Here a_1, a_2, a_3 are the propensity functions for the 3 reactions.

$$g_1 = a_1(\mathbf{x}_1) = a_1(S, E, ES, P) = c_1 \times S \times E \text{ corresponding to } \mathbf{x}_1 = (2, 1, 0, 0)$$

$$g_2 = a_2(\mathbf{x}_2) = a_2(S, E, ES, P) = c_2 \times ES \text{ corresponding to } \mathbf{x}_2 = (1, 0, 1, 0)$$

$$g_3 = a_3(\mathbf{x}_2) = a_3(S, E, ES, P) = c_3 \times ES \text{ corresponding to } \mathbf{x}_2 = (1, 0, 1, 0)$$

$$g_4 = a_1(\mathbf{x}_3) = a_1(S, E, ES, P) = c_1 \times S \times E \text{ corresponding to } \mathbf{x}_3 = (1, 1, 0, 1)$$

$$g_5 = a_2(\mathbf{x}_4) = a_2(S, E, ES, P) = c_2 \times ES \text{ corresponding to } \mathbf{x}_4 = (0, 0, 1, 1)$$

$$g_6 = a_3(\mathbf{x}_4) = a_3(S, E, ES, P) = c_3 \times ES \text{ corresponding to } \mathbf{x}_4 = (0, 0, 1, 1)$$

Integrating CME in this scenario, we get a matrix \mathcal{M} which includes all the possible propensities.

$$\mathcal{M} = \begin{bmatrix} -g_1 & g_2 & 0 & 0 & 0 \\ g_1 & -(g_2 + g_3) & 0 & 0 & 0 \\ 0 & g_3 & -g_4 & g_5 & 0 \\ 0 & 0 & g_4 & -(g_5 + g_6) & 0 \\ 0 & 0 & 0 & g_6 & 0 \end{bmatrix}$$

We use (2.13) to find the probabilities. Choosing $t = 10$ we get the results shown in figure 2.5.

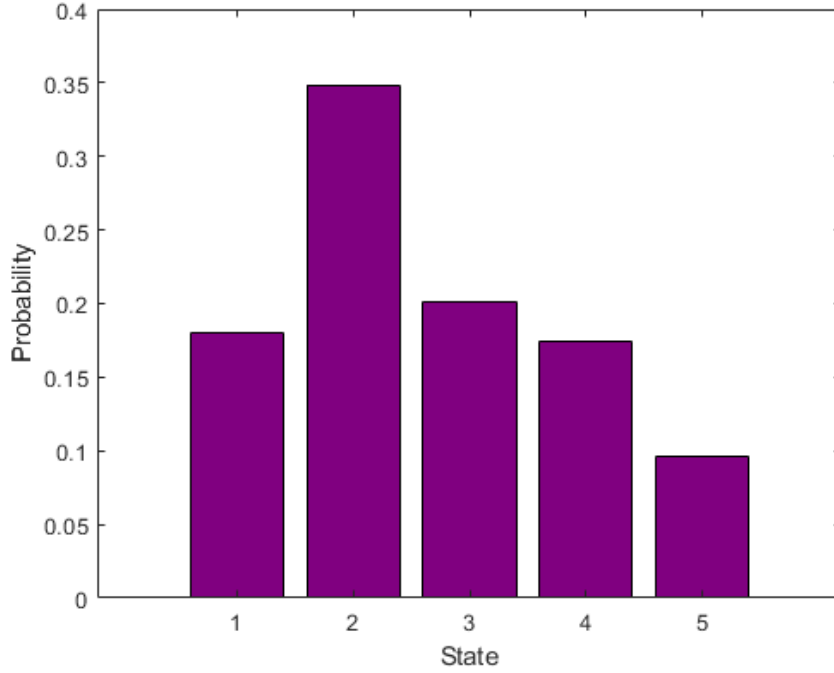


Figure 2.5: Probabilities of all possible states of Michaelis-Menten model with $t = 10$.

2.10 Reaction-Diffusion Master Equation

Assume the domain Ω is partitioned into compartments (voxels). We label the compartments with $V_k, k = 1, \dots, K$. Molecules within each compartment can react with one another within that compartment, and they can also diffuse across the boundaries and move to neighboring compartments. Both the reaction and diffusion processes are considered as random processes. Let, $X_{i,k}(t)$ be the number of molecules of species S_i in compartment V_k at time t . Then each species in the domain is given by the subvector $\mathbf{X}_i(t) = [X_{i,1}(t), \dots, X_{i,K}(t)], i = 1, \dots, N$. That means \mathbf{X}_1 is the subvector which represents the species 1 in all compartments, \mathbf{X}_2 represents species 2, and so on. Thus the state vector of the system is $\mathbf{X} = [\mathbf{X}_1, \dots, \mathbf{X}_N]$. The diffusion propensity function $d_{i,j,k}$ and the state change vector $\boldsymbol{\mu}_{k,j}$ characterize the dynamics of the diffusion of species S_i from compartment V_k to V_j . The vector $\boldsymbol{\mu}_{k,j}$ has a length of K with -1 in the k th position, 1 in the j th position, and 0 elsewhere. Given, $\mathbf{X}(t) = \mathbf{x}$ the diffusion master equation (DME)

can be written by

$$\begin{aligned} \frac{dP(\mathbf{x}, t)}{dt} = & \sum_{i=1}^N \sum_{k=1}^K \sum_{j=1}^K [d_{i,j,k}(\mathbf{x}_i - \boldsymbol{\mu}_{k,j}) P(\mathbf{x}_1, \dots, \mathbf{x}_i - \boldsymbol{\mu}_{k,j}, \dots, \mathbf{x}_N, t) \\ & - d_{i,j,k}(\mathbf{x}_i) P(\mathbf{x}, t)]. \end{aligned} \quad (2.14)$$

The diffusion propensity function $d_{i,j,k}(\mathbf{x}_i) = D/l^2$, for $k = j \pm 1$, where D is the diffusion rate and l is the characteristic length of the compartment. If \mathcal{D} is the transition matrix describing the diffusion of molecules, the equivalent matrix–vector form can be written by

$$\dot{\mathbf{P}}(t) = \mathcal{D} \cdot \mathbf{P}(t). \quad (2.15)$$

Combining equations (2.12) and (2.15) we get the matrix-vector form of the Reaction-Diffusion Master Equation,

$$\dot{\mathbf{P}}(t) = (\mathcal{M} + \mathcal{D}) \cdot \mathbf{P}(t). \quad (2.16)$$

Equation (2.16) is a system of linear constant coefficient ODEs and gives us more possible states than the CME, and so its corresponding $(\mathcal{M} + \mathcal{D})$ is substantially extended to represent species in compartments. Overall, the RDME captures both the reaction and diffusion hence it has a much higher dimensionality.

2.11 Birth-Death Process

The birth-death process [17, 50] is a fundamental stochastic model in molecular biology, used to model the dynamic behavior of biological systems where entities are created (birth) and removed (death) over time. It is particularly useful in modeling population dynamics, biochemical reactions, queuing systems, and epidemiological processes. We work with a

well-studied birth-death process, which can be seen as a simplified model of the production and degradation of a single molecular species [73]. Consider the model for the production and degradation of a specific protein A . Protein molecules can be independently degraded at a constant rate c_1 and be produced at a constant rate c_2 . The reaction scheme is given by

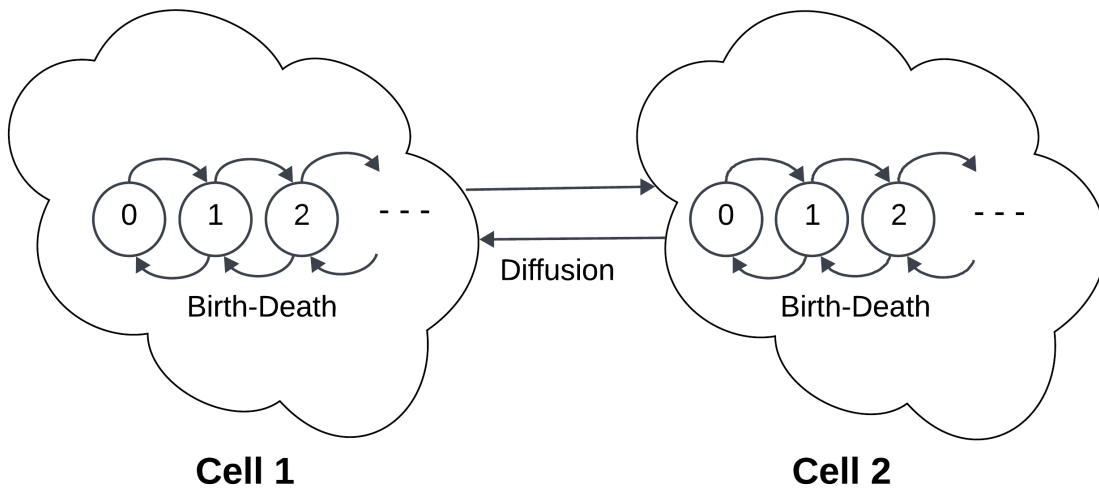


Figure 2.6: Illustration of RDME in birth-death model.

We consider two neighboring cells (i.e. compartments) where protein molecules can move back and forth. We label the cells as Cell 1 and Cell 2. Thus, the system can be represented as a reaction-diffusion process in which reactions (2.17) and (2.18) occur independently inside both cells, while molecules can move between Cell 1 and Cell 2. We model the scheme with RDME consisting of two reactions with two compartments illustrated in figure 2.6. Let A_1 be the number of protein molecules in cell 1, A_2 the number of protein molecules in cell 2. The reactions R_1 through R_4 in table 2.2 reflect (2.17)

and (2.18) inside both cells, while R_5 through R_6 represent movement between cells.

Table 2.2: Reactions, propensities and state change vectors of birth-death model.

Reaction	Propensity	State change vector
$R_1: A_1 \xrightarrow{c_1} \emptyset$	$c_1 \times A_1$	$[-1, 0]$
$R_2: \emptyset \xrightarrow{c_2} A_1$	c_2	$[1, 0]$
$R_3: A_2 \xrightarrow{c_1} \emptyset$	$c_1 \times A_2$	$[0, -1]$
$R_4: \emptyset \xrightarrow{c_2} A_2$	c_2	$[0, 1]$
$R_5: A_1 \xrightarrow{d} A_2$	$d \times A_1$	$[-1, 1]$
$R_6: A_2 \xrightarrow{d} A_1$	$d \times A_2$	$[1, -1]$

Consider that initially, there are 2 protein molecules in cell 1, and 3 protein molecules in cell 2. The initial state vector is $[2, 3]^T$, and we set the reaction parameters $c_1 = .10$ and $c_2 = .20$, and the diffusion parameters $D = 1.0$ and $l = 10$, resulting $d = D/l^2 = 1.0/100 = .01$. The marginal probability distribution of number of molecules in each compartment after time $t = 50$ is shown in the figure 2.7.

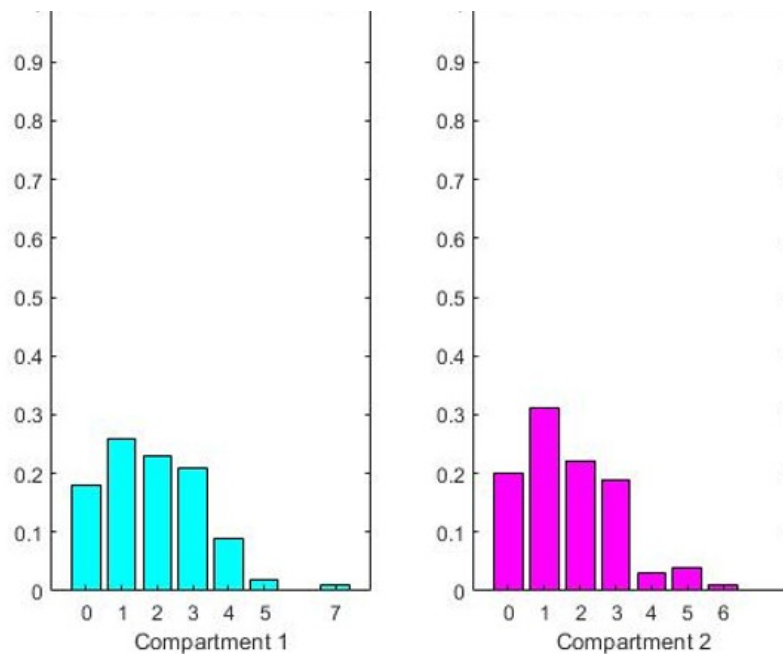


Figure 2.7: Marginal probabilities of the molecules of birth-death model after $t = 50$.

2.12 Spatiotemporal Chemical Master Equation

2.12.1 Metastable Compartments

A metastable compartment refers to a region or state within a system where molecules or particles can temporarily reside before transitioning to a more stable state or being involved in a reaction. This temporary existence occurs due to energy barriers between different states of the system. Despite not being the most energetically favorable state, molecules can linger in this state for a period before moving to a lower energy state. These metastable states can significantly influence the kinetics and dynamics of chemical reactions by affecting reaction pathways, reaction rates, and overall system behavior. For example, in a reaction network, certain intermediate species may be metastable—they are not the initial reactants nor the final products, but they are transiently formed during the course of the reaction before ultimately transforming into stable end products.

2.12.2 Particle Based Reaction-Diffusion

Particle-based reaction-diffusion dynamics (PBRD) is a computational approach used to simulate the spatiotemporal behavior of particles undergoing diffusion and chemical reactions within a given environment. In PBRD simulations, individual particles, representing molecules or other entities of interest, are tracked as they move and interact with each other based on predefined rules governing diffusion and reaction kinetics. At its core, PBRD involves two main processes: diffusion and reaction. Diffusion refers to the random movement of particles due to thermal energy, which can lead to the spreading or dispersal of substances throughout the system. Reaction involves the transformation of particles when they encounter each other, leading to chemical reactions and the creation or destruction of different molecular species.

2.12.3 ST-CME

The spatiotemporal chemical master equation (ST-CME) is a mathematical framework used to model the dynamics of chemical reactions occurring in spatially extended systems

over time. It extends the traditional chemical master equation, which describes the temporal evolution of the probabilities of different molecular species in a well-mixed system, to include spatial dependencies. In many biological and chemical systems, spatial heterogeneity plays a crucial role in determining system behavior. The ST-CME accounts for spatial variations in concentrations, diffusion, and reaction rates, allowing for a more accurate description of reaction kinetics in spatially extended systems. Consider $N_r^l(t)$ be the number of particles of species l in compartment r at time t . Then

$\mathbf{N}_r(t) = (N_r^1(t), \dots, N_r^L(t))$ denotes the states of present species in compartment r . All the state of the system at time t is given by the matrix $\mathbf{N}(t) = (\mathbf{N}_r(t))_{r=1, \dots, M} \in \mathbb{N}_0^{M, L}$. Here, \mathbf{E}_r^l is a matrix whose elements are all zero except the entry (r, l) which is one. Let λ_{rs}^l denote the jump rate for each individual particle of species l . Then the total probability per unit of time for a jump of species l from compartment r to s at time t is given by $\lambda_{rs}^l N_r^l(t)$. For each of the reactions R_k , suppose $\boldsymbol{\nu}_k = (\nu_k^1, \dots, \nu_k^L)$ describe the change in the number of copies of all species induced by this reaction. Reaction R_k occurring in the r th compartment refers to the transition $\mathbf{N}_r(t) \rightarrow \mathbf{N}_r(t) + \boldsymbol{\nu}_k$. In order to specify where the reaction takes place, the vector $\boldsymbol{\nu}_k$ is multiplied by a column vector \mathbf{e}_r with the value 1 at entry r and zeros elsewhere. This gives a matrix $\mathbf{e}_r \boldsymbol{\nu}_k$ whose r th row is equal to $\boldsymbol{\nu}_k$ while all other rows contain zeros. With these settings, let $P(\mathbf{n}, t)$ be the probability that the process is in state \mathbf{n} at time t given an initial state $\mathbf{N}(0) = \mathbf{n}_0$. The ST-CME is then given by,

$$\begin{aligned} \frac{dP(\mathbf{n}, t)}{dt} = & \sum_{r=1}^M \sum_{s \neq r} \sum_{l=1}^L (\lambda_{sr}^l (n_s^l + 1) P(\mathbf{n} + \mathbf{E}_s^l - \mathbf{E}_r^l, t) - \lambda_{rs}^l n_r^l P(\mathbf{n}, t)) \\ & + \sum_{r=1}^M \sum_{k=1}^K (\alpha_r^k (\mathbf{n} - \mathbf{e}_r \boldsymbol{\nu}_k) P(\mathbf{n} - \mathbf{e}_r \boldsymbol{\nu}_k, t) - \alpha_r^k (\mathbf{n}) P(\mathbf{n}, t)). \end{aligned}$$

2.12.4 Binding and Unbinding

An eukaryotic cell is a cell that possesses a membrane-bound nucleus and other membrane-bound organelles. They are characterized by their complex internal structure

and compartmentalization, which allows for specialized functions to occur within different organelles.

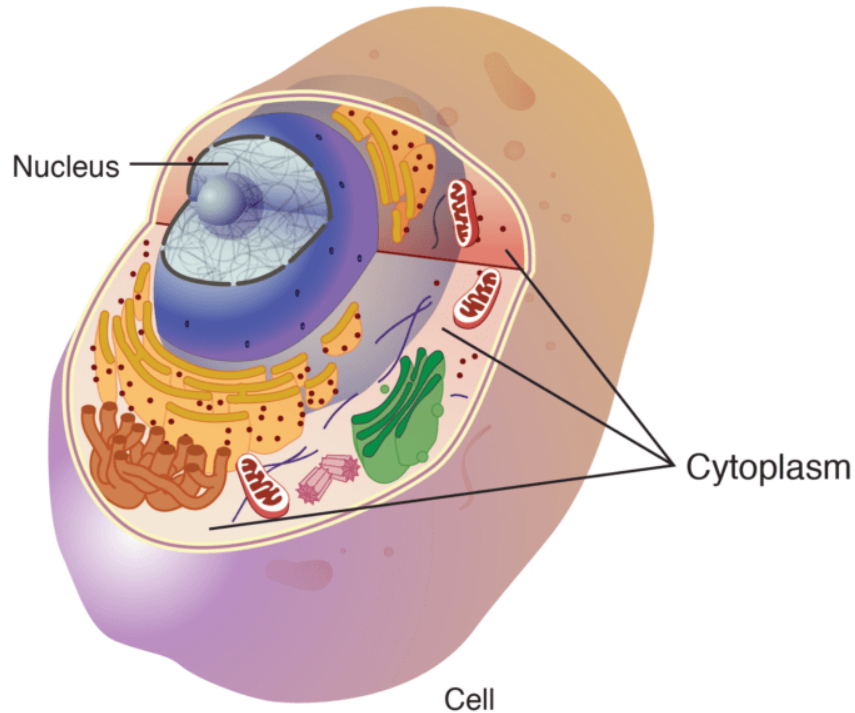
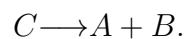
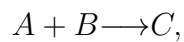


Figure 2.8: Eukaryotic cell (image from genome.gov).

Consider a very general system consisting of three species A , B , and C with the binding and unbinding reactions given by,



We assume that the binding reaction takes place only in the cytoplasm while unbinding is restricted to the nucleus. This separation of reactions, combined with the metastability of diffusion due to a reduced permeability of the nuclear membrane, obviously keeps the process from being well-mixed in total space. Consider, there are 5 A -particles uniformly distributed in the nucleus and 5 B -particles uniformly distributed in the cytoplasm. The average evolution of the population over time is shown in figure 2.9, 2.10, and 2.11.

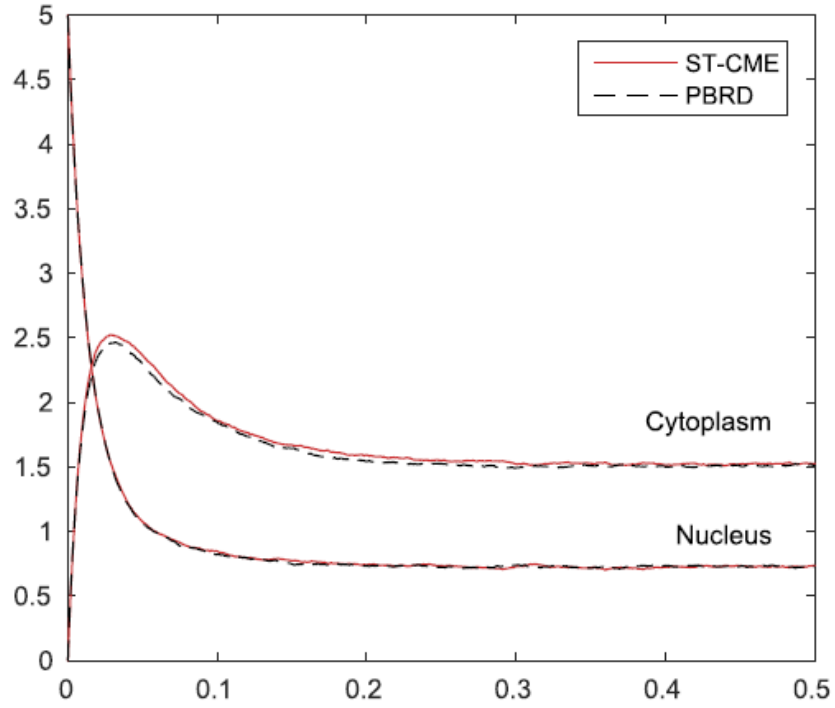


Figure 2.9: Evolution of species A (image adapted from [74]).

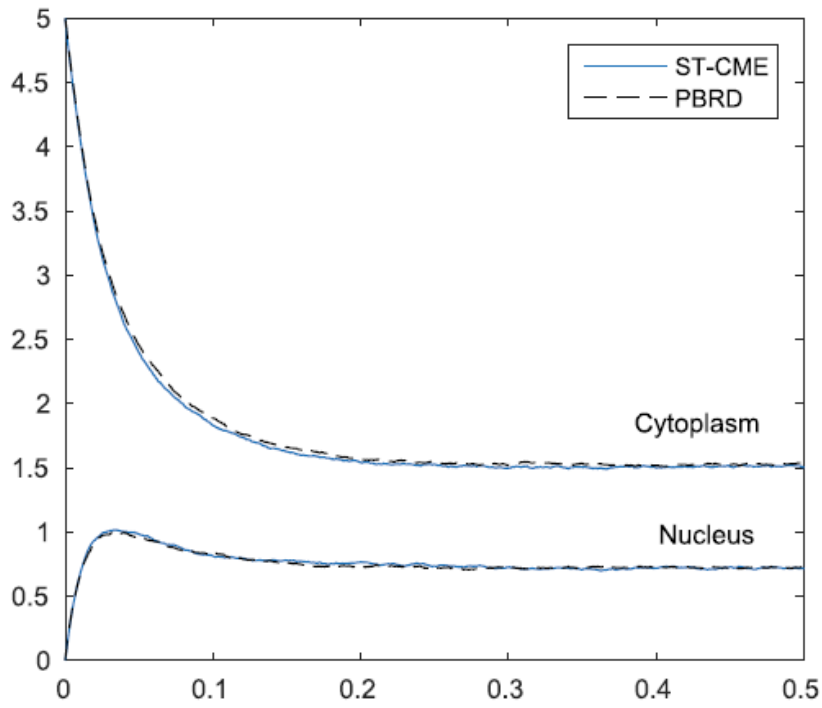


Figure 2.10: Evolution of species B (image adapted from [74]).

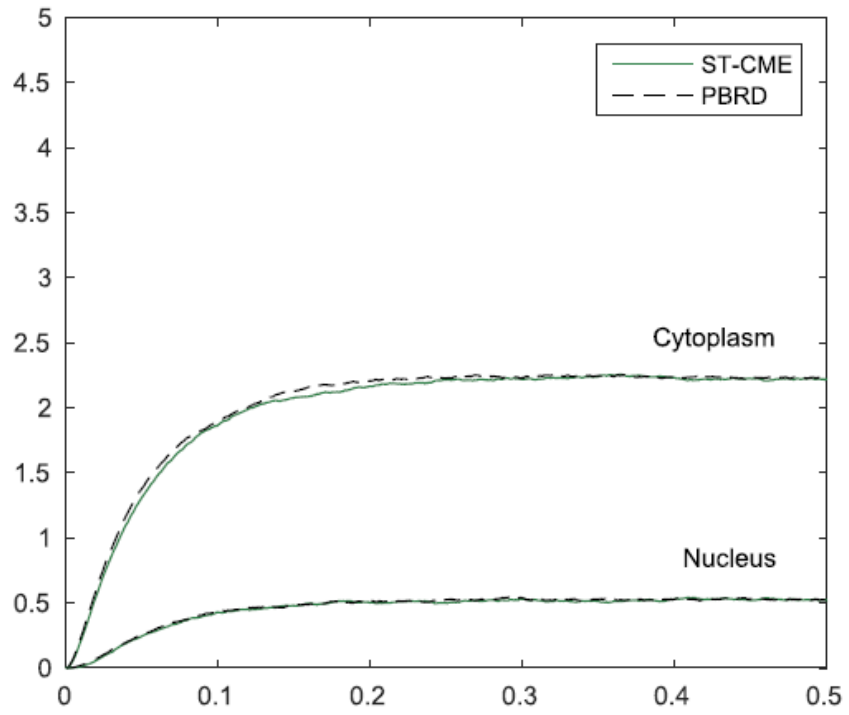


Figure 2.11: Evolution of species C (image adapted from [74]).

CHAPTER 3 A STUDY OF A METAPOPOPULATION MODEL USING THE STOCHASTIC REACTION-DIFFUSION MASTER EQUATION

This chapter is based on work published in Springer book, SDSC 2023. Here, we formulate a metapopulation model using the RDME. Our study employs the FSP technique to the metapopulation model. Although the RDME model is similar with the CME, it is computationally more expensive since the state space becomes substantially larger. We have applied the SSA algorithm to analyze the marginal distribution of the species involved.

3.1 Stochastic Simulation Algorithm of Gillespie

The "curse of dimensionality" obstructs finding an analytical solution to the CME for larger systems. This arises because the number of possible states increases exponentially as the number of molecular species and reactions increases. To overcome this limitation, Gillespie's SSA [31] is widely used as a numerical method to simulate the time evolution of chemical reaction networks. This method is based on the principles of the CME and generates a stochastic trajectory of the network by simulating individual reactions one by one. SSA selects the next reaction and time interval by calculating the propensity functions of all possible reactions based on the current state of the system, generating a random number to determine the time until the next reaction occurs, and selecting the next reaction based on probabilities of each reaction occurring. The algorithm updates the system state according to the reaction stoichiometry and repeats the process.

The SSA produces a pair of random numbers, namely r_1 and r_2 , during every individual step. Note that r_1 and r_2 both belong to $U(0, 1)$, which is the set of uniformly distributed

random numbers in the interval $(0, 1)$. The time for the next reaction to occur is given by $t + \tau$, where τ is given by

$$\tau = \frac{1}{a_0} \ln \left(\frac{1}{r_1} \right). \quad (3.1)$$

It is now important to know which reaction will occur next. The index λ of the next reaction is given by the smallest integer which satisfies the following inequality,

$$\sum_{j=1}^{\lambda} a_j > r_2 a_0, \quad (3.2)$$

where,

$$a_0(\mathbf{x}) = \sum_{j=1}^M a_j(\mathbf{x}). \quad (3.3)$$

The states of the system are updated by $\mathbf{X}(t + \tau) = \mathbf{X}(t) + \boldsymbol{\nu}_\lambda$. Subsequently, the simulation proceeds to the time of the next reaction. By generating a large number of stochastic trajectories, the SSA method can provide an accurate approximation of the behavior of the reaction network over time, even in cases where an analytical solution is not feasible due to the curse of dimensionality.

3.2 Finite State Projection

Instead of simulating an ensemble of trajectories using SSA, the FSP method [57] directly computes an analytical approximation to the solution of the CME. This is achieved by creating a computationally manageable projection of the full state space and calculating the time evolution of the probability mass function within this projection space. The FSP technique uses a truncated state space to solve the CME and estimate the probability vector (PV) of the populations in a chemical reaction system. Let $T = \{1, \dots, k\}$, then the transition matrix of (2.12) is replaced by \mathcal{M}_T where

$$\mathcal{M} = \left(\begin{array}{c|c} \mathcal{M}_T & * \\ \hline * & * \end{array} \right) \in \mathbb{R}^{n \times n}.$$

Here, $k = |T|$ is the cardinality of T and \mathcal{M}_T is a $k \times k$ submatrix of the true operator \mathcal{M} . The finite state projection is then formed by the states indexed by T . For an initial truncated PV denoted by $\mathbf{P}_T(t=0)$, the FSP finds the PV at any time $\mathbf{P}_T(t)$, by the following,

$$\dot{\mathbf{P}}_T(t) = \mathcal{M}_T \cdot \mathbf{P}_T(t). \quad (3.4)$$

Equation (3.4) is a system of linear ODEs and the solution is given by

$$\mathbf{P}_T(t) = \exp(\mathcal{M}_T t) \mathbf{P}_T(0). \quad (3.5)$$

Observe that, the subscript T characterizes the truncation just mentioned. In addition, the initial distribution is truncated in the same way. \mathcal{M}_T needs not merely be a principal submatrix. Instead, assume that T is an arbitrary subset of $\{1, \dots, n\}$ and for clarity of presentation \mathcal{M}_T is defined with the same size as the matrix $\mathcal{M} = [m_{ij}]$ using

$$(\mathcal{M}_T)_{ij} = \begin{cases} m_{ij} & \text{if } i, j \in T \\ 0 & \text{if } i \notin T \text{ or } j \notin T \end{cases}$$

Similarly, \mathbf{P}_T is defined from $\mathbf{P} = (p_1, \dots, p_n)^T$ using

$$(\mathbf{P}_T)_i = \begin{cases} p_i & \text{if } i \in T \\ 0 & \text{otherwise} \end{cases}$$

3.3 Metapopulation Model

Metapopulation models [1, 4, 14], which are used to study populations that are divided into subpopulations connected by migration, often employ reaction-diffusion models to describe spatial dynamics [13]. In a reaction-diffusion metapopulation model, there is a diffusion term and a reaction term. The diffusion term captures the movement of individual between neighboring subpopulations, while the reaction term describes birth and death rates within each subpopulation. The reaction-diffusion model has many applications, such as predicting how changes in habitat quality or connectivity may affect a species' distribution and abundance, studying disease spread, and assessing the persistence of rare or endangered species [6, 26]. By taking into account both diffusion and reaction terms, the model can better capture complex interactions between local population dynamics and spatial dispersal.

In a metapopulation model using the RDME, each subpopulation's population is modeled as a stochastic process with birth, death, and migration events occurring randomly. The model includes diffusion of individuals between subpopulations and reactions within each subpopulation, such as birth and death rates. The RDME takes into account the stochasticity of these events, allowing for more accurate predictions of population dynamics compared to deterministic models.

For the purpose of experimentally demonstrating a proof-of-concept in this study, we consider a basic reaction scheme conserving the number of particles



This system has been widely used both in physics and mathematical epidemiology. For its interpretation with the lenses of our metapopulation model, it would be that α particles represent normal particles (healthy individuals) and β particles represent active particles

(infected individuals) intermingling from area to area. This scheme is also known as the SIS model [2]. We consider two neighboring areas (i.e., compartments) where both the healthy and infected individuals can move back and forth. We label the areas as Area 1 and Area 2. Hence the scheme can be represented by a reaction-diffusion process where the reactions (3.6) and (3.7) can take places inside both areas separately and the individuals can move back and forth between Area 1 and Area 2. An illustration is shown in figure 3.1

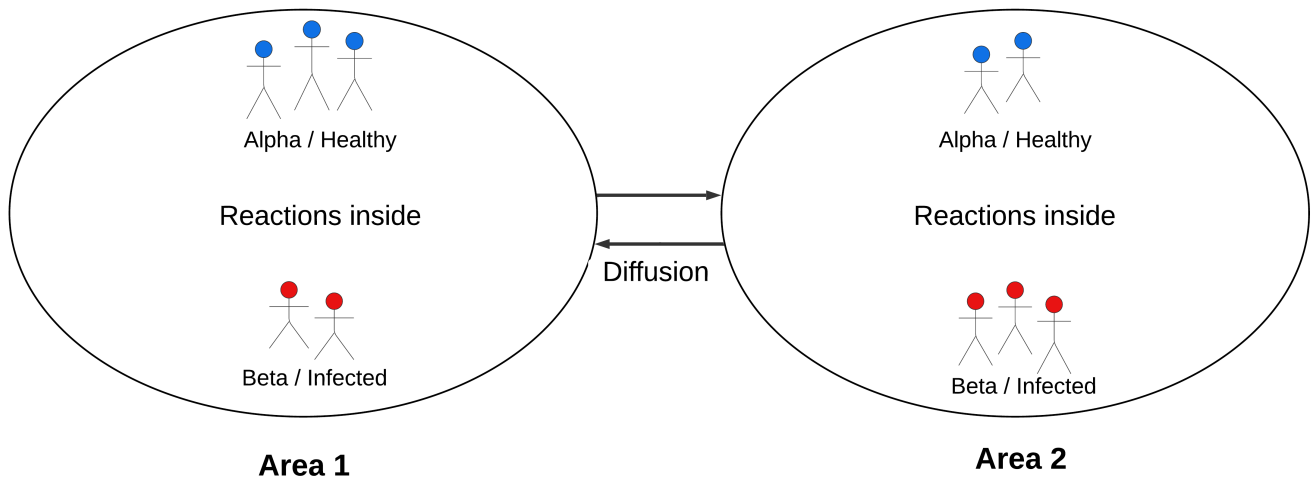


Figure 3.1: Illustration of RDME in metapopulation model.

3.4 Integrating RDME into Metapopulation Model

The process involved in the metapopulation model is a random process and can be modeled by the RDME consisting of two reactions with two compartments [62]. Let α_1 be the number of healthy individuals in area 1, β_1 the number of infected individuals in area 1, α_2 the number of infected healthy in area 2, β_2 the number of infected individuals in area 2. The reactions R_1 through R_4 in table 3.1 reflect (3.6) and (3.7) inside both areas, while R_5 through R_8 represent diffusion between areas.

Table 3.1: Reactions, propensities and state change vectors of metapopulation model.

Reaction	Propensity	State change vector
$R_1: \beta_1 \xrightarrow{c_1} \alpha_1$	$c_1 \times \beta_1$	$[1, -1, 0, 0]$
$R_2: \beta_1 + \alpha_1 \xrightarrow{c_2} 2\beta_1$	$c_2 \times \beta_1 \times \alpha_1$	$[-1, 1, 0, 0]$
$R_3: \beta_2 \xrightarrow{c_1} \alpha_2$	$c_1 \times \beta_2$	$[0, 0, 1, -1]$
$R_4: \beta_2 + \alpha_2 \xrightarrow{c_2} 2\beta_2$	$c_2 \times \beta_2 \times \alpha_2$	$[0, 0, -1, 1]$
$R_5: \alpha_1 \xrightarrow{d} \alpha_2$	$d \times \alpha_1$	$[-1, 0, 1, 0]$
$R_6: \beta_1 \xrightarrow{d} \beta_2$	$d \times \beta_1$	$[0, -1, 0, 1]$
$R_7: \alpha_2 \xrightarrow{d} \alpha_1$	$d \times \alpha_2$	$[1, 0, -1, 0]$
$R_8: \beta_2 \xrightarrow{d} \beta_1$	$d \times \beta_2$	$[0, 1, 0, -1]$

To allow visualizing our prototype, we assume that initially, there are 2 healthy individuals and 1 infected individual in area 1. Similarly, in area 2 there are 1 healthy and 2 infected individuals. The initial state vector is $[2, 1, 1, 2]^T$, and we set the reaction parameters $c_1 = .30$ and $c_2 = 1.0$, and the diffusion parameters $D = 1.0$ and $l = 10$. The RDME gives us all the possible states we can get when those individuals react and diffuse randomly. Even such a simple initial state generates 84 possible states.

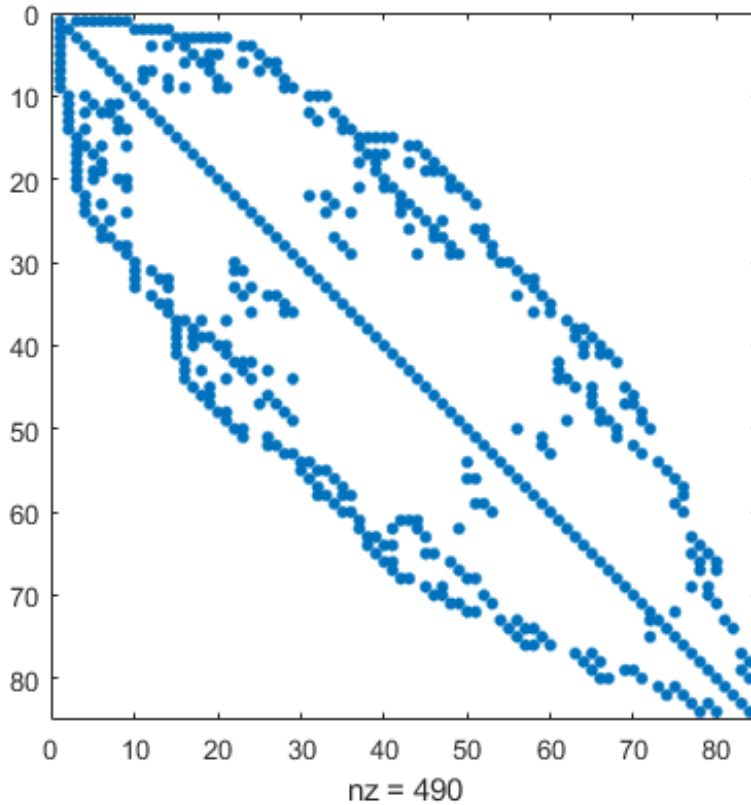


Figure 3.2: Visualization of the sparsity pattern of the transition matrix in RDME.

Figure 3.2 shows the sparsity pattern of the transition matrix involved in the RDME formulation where the transition matrix covers both reaction and diffusion process. In the CME, the transition matrix would have only been responsible for the reactions. In the RDME the matrix we have is a 84×84 matrix, where the dots in the figure 3.2 represent the 490 non-zero elements of the transition matrix and the other parts without dots indicate zeros. As noted earlier, this example is a proof-of-concept with a resulting figure that a reader can easily visualize. More realistic models lead to substantially larger sizes. Indeed, for a problem with N species S_i that each can attain 0 to $n_i - 1$ copies, the CME matrix can already be of size $\prod_{i=1}^N n_i \times \prod_{i=1}^N n_i$, so that if $N = 10$ and $n_i \approx 10$ the size would be a huge $10^{10} \times 10^{10}$ for the CME and even more than that for the RDME.

3.5 Numerical Results

We solve the RDME using the FSP equation (3.5) by truncating the state space to 70, to approximate the probability of the states. We are particularly interested to get the marginal probability of the individuals. By definition, the marginal probability of an event is the likelihood of an event happening regardless of the outcome of the other events. Assuming $t = 10$, we approximate the marginal probability distribution of the healthy and infected individuals in two areas. The results are shown in figure 3.3.

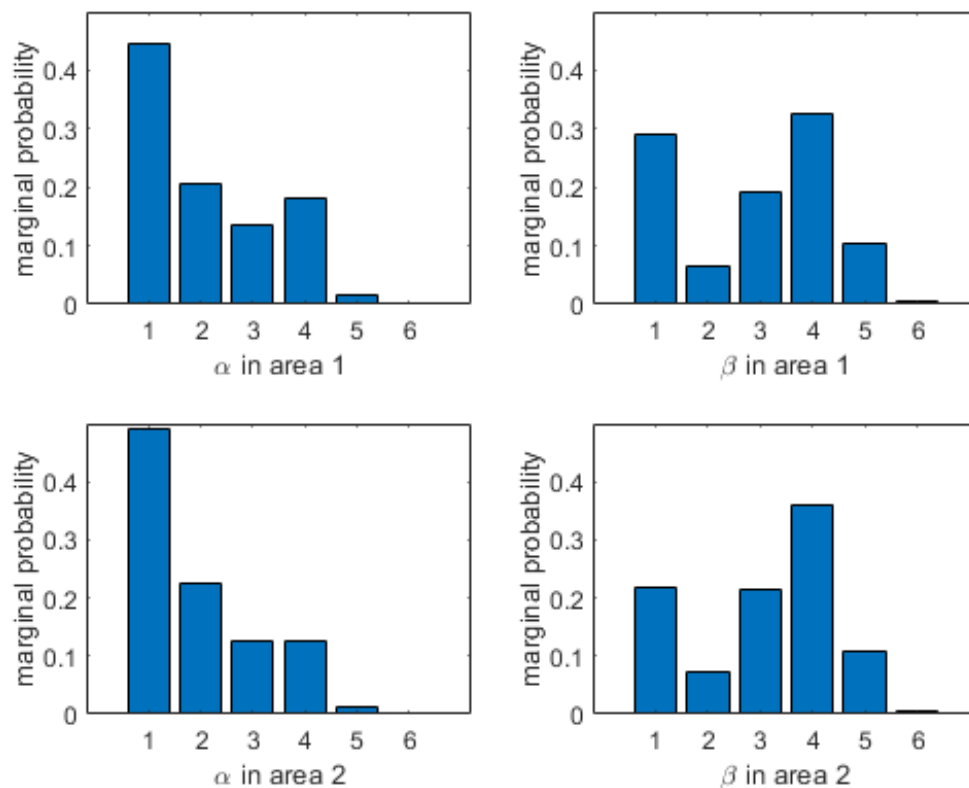


Figure 3.3: Marginal probability distribution of the number of healthy and infected individuals in two areas using FSP at time $t = 10$.

Another algorithm of analyzing the model is using the SSA. Contrary to the FSP, the SSA generates realizations. The computational time of this algorithm depends on the number of realizations taken into account. Here we have taken 100 realizations to find the marginal probability distribution of the individuals, with the result shown in figure 3.4.

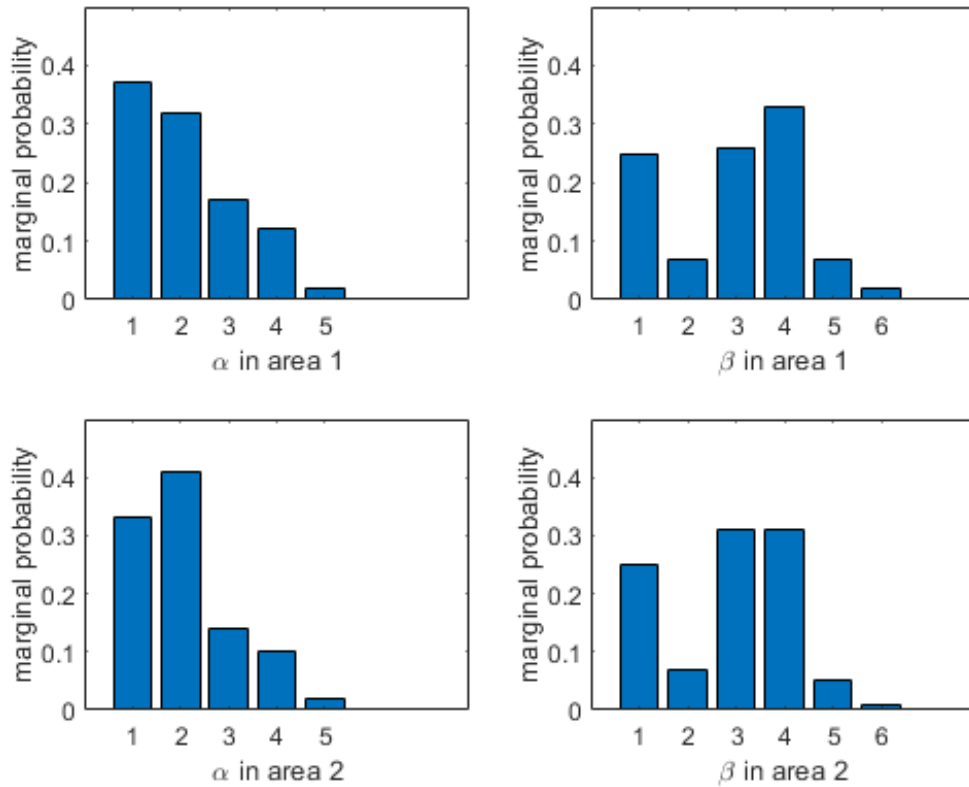


Figure 3.4: Marginal probability distribution of the number of healthy and infected individuals in two areas using SSA at time $t = 10$.

3.6 Discussion and Conclusion

We have employed the RDME formulation in a metapopulation model that encompasses both reaction and diffusion processes and we have used FSP and SSA to find the marginal probability distribution. The FSP works with a truncated state space instead of considering all the possible states. And when the copy numbers are low the SSA-results can be subject to statistical noise. These are the reasons for the differences between the results shown in figure 3.3 and figure 3.4. The RDME approach is suitable for analyzing the model, given its ability to relate the model, even with the challenge of the "curse of dimensionality". The curse of dimensionality refers to the fact that as the dimensionality of the model increases, the number of possible states grows exponentially, making it

increasingly difficult to calculate the probabilities of future states. In addition, implementing the FSP in the RDME or using the SSA algorithm to approximate the probability of future states can prove to be computationally expensive. In cases where the metapopulation model is more complex and involves an exceedingly vast state space, the computational time required to solve it using either of these methods becomes prohibitively high. As such, the main objective of our future work is to overcome these challenges and find a more efficient way to solve the RDME for advanced metapopulation models. To achieve this goal, we envision the use of tensor train decomposition techniques [59]. The basic idea is to represent a high-dimensional tensor as a sequence of smaller tensors that are multiplied together. This sequence of smaller tensors is called the tensor train. Each tensor in the sequence has a fixed size and is connected to the neighboring tensors by a set of indices. Applications of tensors already exist in the context of the CME [21, 49], and we similarly envision that they could provide an alternative approach to calculate the RDME solution that can significantly reduce computational time while still maintaining accuracy, thereby enabling us to meaningfully analyze and model metapopulation systems and gain insights into their dynamics and behavior. Overall, a successful implementation of the tensor train decomposition technique in the context of the RDME formulation for metapopulation modeling could pave the way for more efficient and accurate modeling of complex systems. This, in turn, could have significant applications in various fields such as ecology, biology, and epidemiology. By improving our understanding of these systems, we can better predict and mitigate the spread of diseases, preserve ecological systems, and ensure the long-term sustainability of biological populations.

CHAPTER 4 AN APPLICATION OF TENSORS IN THE STOCHASTIC REACTION-DIFFUSION MASTER EQUATION

Tensors, fundamentally, are extensions of vectors, and matrices, with the ability to represent and manipulate data in multiple dimensions. Tensor methods, also known as tensor approaches, encompass the utilization of tensors for the purpose of problem-solving, data analysis, and the extraction of significant patterns from datasets with many dimensions. The concept of tensors and their decompositions was first introduced in the year 1927 [45]. Tensors find utility in several machine learning applications, such as facial recognition, analysis of musical scores, and identification of cliques within social networks [60, 70]. An overview of tensor techniques for large-scale numerical computations is given in [36, 38], geared towards a scientific computing audience.

In this chapter, we addressed RDME using tensor methods. We recall the RDME framework in a metapopulation model. Despite having similarities to the CME, the RDME model requires more computing power because of the significantly wider state space. To obtain the marginal distribution of the pertinent species, we apply the tensor technique. We've also used the FSP method with the metapopulation model for comparison purpose.

4.1 Matrix Products

Defined here are additional operations on matrices not as well known as addition and multiplication that will be required in the upcoming sections.

4.1.1 Hadamard Product

Consider two matrices A and B . The Hadamard product is denoted $A * B$ and represents the element-wise product. We can only find the Hadamard product when the size of two matrices are same.

4.1.2 Kronecker Product

The Kronecker product of two matrices A (size $I \times J$) and B (size $K \times L$) is expressed by $A \otimes B$ and the resulting matrix is of size $(IK) \times (JL)$. Mathematically, the Kronecker product is shown below,

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1J}B \\ a_{21}B & a_{22}B & \dots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \dots & a_{IJ}B \end{bmatrix}$$

4.1.3 Khatri-Rao Product

This is another useful matrix operation which incorporates the Kronecker product. The Khatri-Rao product of two matrices A (size $I \times J$) and B (size $K \times J$) is denoted by $A \odot B$ and the resulting matrix is of size $(IK) \times (J)$.

$$A \odot B = \begin{bmatrix} a_1 \otimes b_1 & a_2 \otimes b_2 & \dots & a_J \otimes b_J \end{bmatrix}$$

4.2 Tensor Notations and Definitions

4.2.1 Tensor

A tensor is a multidimensional array. Mathematically, an N -way or N th-order tensor of size $I_1 \times I_2 \times \dots \times I_N$, denoted by $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$, is a set of real numbers $\mathcal{X}(i_1, i_2, \dots, i_N)$, where $i_s = 1 : I_s$, for $s = 1 : N$. Scalars are represented using lowercase letters, while vectors, which are first-order tensors, are indicated by bold lowercase letters. Matrices, corresponding to second-order tensors, are denoted by bold capital letters. For tensors of third order or higher, bold Euler script letters are used. Figure 4.1 represents the visualization of tensors with respect to their orders.

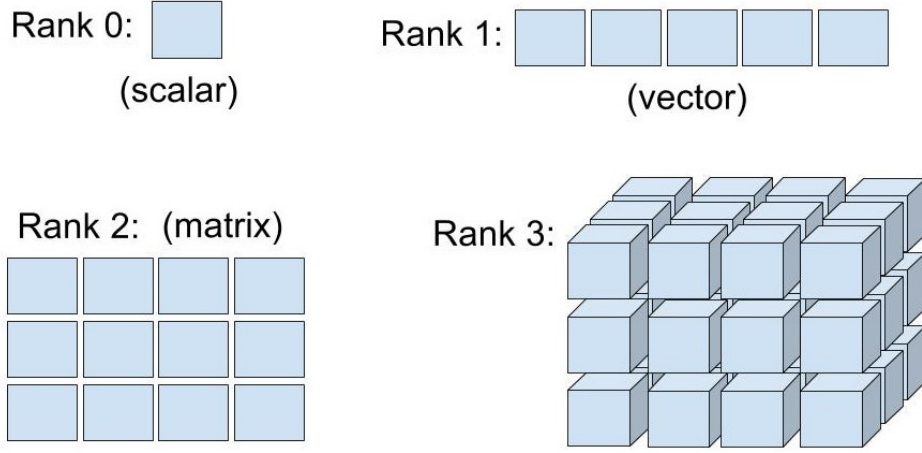


Figure 4.1: Visualization of different tensors (image from packtpub.com).

4.2.2 Subtensor

Consider a N -way tensor, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$. Given $\mathbf{L} = (L_1, L_2, \dots, L_N)$, $\mathbf{R} = (R_1, R_2, \dots, R_N)$, such that $\mathbf{1} \leq \mathbf{L} \leq \mathbf{R} \leq \mathbf{I}$, i.e., $1 \leq L_s \leq R_s \leq I_s, s = 1, \dots, N$. With mode sizes $J_s = R_s - L_s + 1, \mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \dots \times J_N}$, defined by

$$\mathcal{Y}(j_1, j_2, \dots, j_N) = \mathcal{X}(j_1 + L_1 - 1, j_2 + L_2 - 1, \dots, j_N + L_N - 1),$$

is called a subtensor of \mathcal{X} , labeled by $\mathcal{Y} = \mathcal{X}(\mathbf{L} : \mathbf{R})$.

When $L_s = R_s$, for some $s = 1 : N$, we have $J_s = 1$ and the subtensor $\mathcal{Y} = \mathcal{X}(\mathbf{L} : \mathbf{R})$ can be interpreted as a $(N - 1)$ -th way tensor, $\mathcal{Y} \in \mathbb{R}^{J_1 \times \dots \times J_{s-1} \times J_{s+1} \times \dots \times J_N}$ with

$$\begin{aligned} \mathcal{Y}(j_1, \dots, j_{s-1}, j_{s+1}, \dots, j_N) = \\ \mathcal{X}(j_1 + L_1 - 1, \dots, j_{s-1} + L_{s-1} - 1, L_s, j_{s+1} + L_{s+1} - 1, \dots, j_N + L_N - 1). \end{aligned}$$

4.2.3 Fiber

Let $(i_1, i_2, \dots, i_N) \in (1 : I_1) \times (1 : I_2) \times \dots \times (1 : I_N)$. For each $n = 1 : N$, the subtensor $\mathcal{Y} = \mathcal{X}(\mathbf{L} : \mathbf{R})$ where $L_s = R_s = i_s$, for all $s \neq n$, and $L_n = 1, R_n = I_n$, is defined by a mode- n fiber of \mathcal{X} , and can be interpreted as a vector, $\mathcal{Y} = (y_j) \in \mathbb{R}^{I_n}$ with $y_j = x_{i_1 \dots i_{n-1} j i_{n+1} \dots i_N}$, for $j = 1 : I_n$. We can also find mode- n fiber as a vector resulting

from \mathcal{X} by fixing all indices except n . Figure 4.2 illustrates mode-1, mode-2, and mode-3 fibers of a three way tensor.

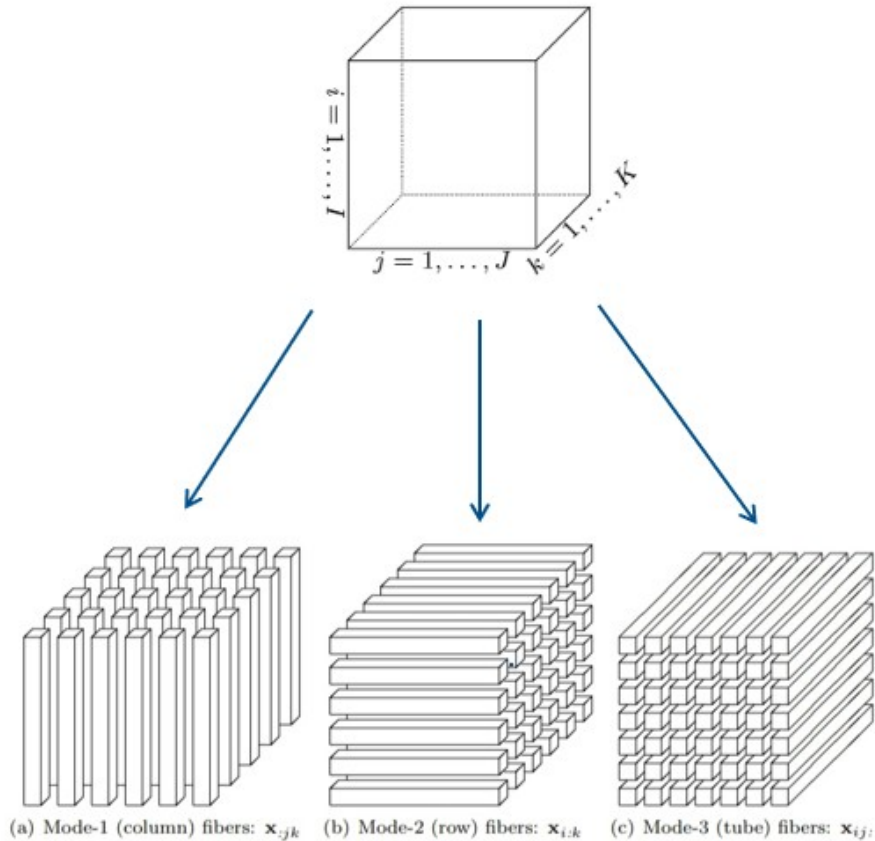


Figure 4.2: Different fibers of a 3-mode tensor.

4.2.4 Slice

For each $m, n = 1 : N, m < n$, the subtensor $\mathcal{Y} = \mathcal{X}(\mathbf{L} : \mathbf{R})$ where $L_s = R_s = i_s$, for all $s \neq m, n$, and $L_m = L_n = 1, R_m = I_m, R_n = I_n$, is called a slice of \mathcal{X} , and can be interpreted as a matrix, $\mathbf{Y} = (y_{jk}) \in \mathbb{R}^{I_m \times I_n}$ with $y_{jk} = x_{i_1 \dots i_{m-1} j i_{m+1} \dots i_{n-1} k i_{n+1} \dots i_N}$, for $j = 1 : I_m$, $k = 1 : I_n$. By fixing all other indices except two, slices of a tensor can be regarded as a matrix arising from \mathcal{X} . Figure 4.3 illustrates three different slices of a 3 way tensor.

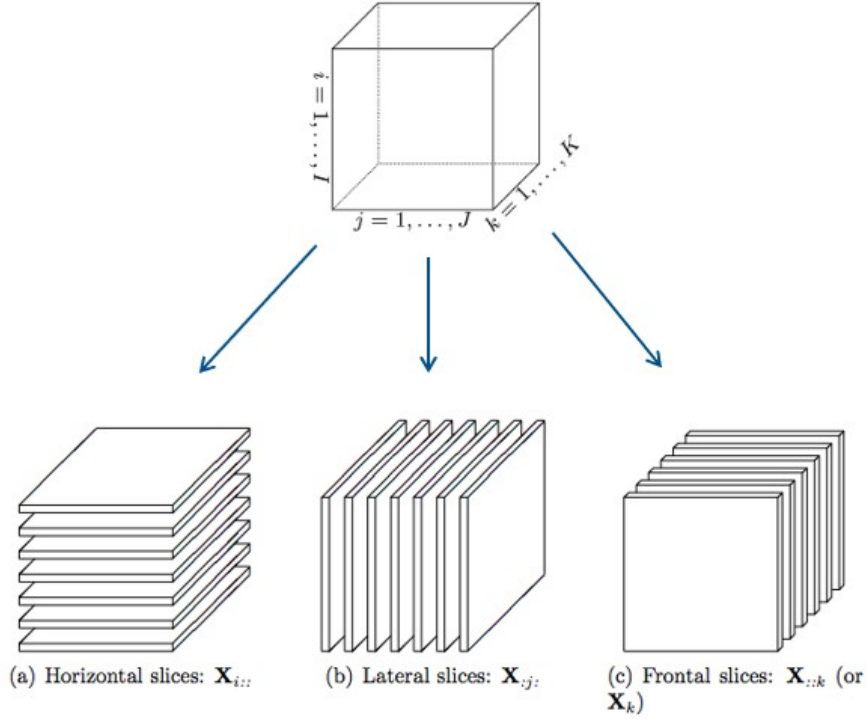


Figure 4.3: Different slices of a 3-mode tensor.

4.2.5 n -rank of a Tensor

The n -rank of \mathcal{X} , represented as $R_n = \text{rank}_n(\mathcal{X})$ is the dimension of the vector space spanned by the mode- n fibers of \mathcal{X} . Another approach to consider the n -rank of a tensor is the dimension of the vector space spanned by the mode- n unfolding of \mathcal{X} .

4.2.6 The reshape and vec Operations on Tensors

Consider the mapping $\text{col}(:, I_1, \dots, I_N) : (1 : I_1) \times \dots \times (1 : I_N) \rightarrow (1 : I_1 \cdot \dots \cdot I_N)$ defined by

$$\text{col}(i_1, \dots, i_N; I_1, \dots, I_N) = i_1 + (i_2 - 1)I_1 + (i_3 - 1)I_1 \cdot I_2 + \dots + (i_N - 1)I_1 \cdot I_2 \cdot \dots \cdot I_{N-1}. \quad (4.1)$$

It can be shown that, for each $k \in (1 : I_1 \cdot \dots \cdot I_N)$, $\text{col}(i_1, \dots, i_N; I_1, \dots, I_N) = k$ if and

only if

$$i_N = (k - 1) \operatorname{div} (I_1 \cdot I_2 \cdot \dots \cdot I_{N-1}), \quad (4.2)$$

and

$$\operatorname{col}(i_1, \dots, i_{N-1}; I_1, \dots, I_{N-1}) = (k - 1) \operatorname{mod} (I_1 \cdot I_2 \cdot \dots \cdot I_{N-1}) + 1. \quad (4.3)$$

Therefore, for each fixed $(I_1, \dots, I_N) \in \mathbb{N}^N$, the relations (4.1), (4.2), and (4.3) can be used to produce the mapping $\operatorname{col}: (1 : I_1) \times \dots \times (1 : I_N) \rightarrow (1 : I_1 \cdot \dots \cdot I_N)$ and its inverse $\operatorname{col}^{-1}: (1 : I_1 \cdot \dots \cdot I_N) \rightarrow (1 : I_1) \times \dots \times (1 : I_N)$.

Using these mapping, the vec operator transforms a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ into the vector $\mathbf{x} = \operatorname{vec}(\mathcal{X}) \in \mathbb{R}^{I_1 \cdot \dots \cdot I_N}$, defined by

$$x_k = \mathcal{X}(\operatorname{col}^{-1}(k; I_1, \dots, I_N)), \text{ for } k = 1 : I_1 \cdot \dots \cdot I_N.$$

and the $\operatorname{reshape}$ operator transforms a vector $\mathbf{x} \in \mathbb{R}^{I_1 \cdot \dots \cdot I_N}$ into the tensor $\mathcal{X} = \operatorname{reshape}(\mathbf{x}) \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$, defined by

$$\mathcal{X}(i_1, \dots, i_N) = x_{\operatorname{col}(i_1, \dots, i_N; I_1, \dots, I_N)}, \text{ for } (i_1, \dots, i_N) \in (1 : I_1) \times \dots \times (1 : I_N).$$

Since the set of all column vectors of \mathbf{X}_n is equal to the set of all mode- n fibers of \mathcal{X} , using the definition of tensor rank,

$$R_n = \operatorname{rank}_n(\mathcal{X}) = \operatorname{rank}(\mathbf{X}_n) \leq I_n.$$

4.3 Tensor Decomposition

Tensor decomposition is a mathematical method employed to decompose tensors into more elementary components, thereby uncovering concealed structures and patterns. It

extends matrix factorization to higher dimensions and is extensively utilized in computational biology, machine learning, signal processing, and data mining. We discuss some of the common tensor decompositions [19, 21, 35, 36, 40, 48, 49, 51–53, 58, 59] in the upcoming subsections.

4.3.1 Higher Order Singular Value Decomposition (HOSVD)

Singular Value Decomposition (SVD) is a fundamental matrix factorization method that represents any matrix as the product of two orthogonal matrices plus a diagonal matrix of singular values, encapsulating the inherent geometric characteristics of the data. Higher Order Singular Value Decomposition (HOSVD) generalizes this concept to multi-dimensional arrays, or tensors, by decomposing them into a core tensor and a collection of orthogonal matrices corresponding to each mode. Here, we present some properties of HOSVD [19],

Theorem 4.3.1.1. *Every tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ can be written as*

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \dots \times_N \mathbf{A}^{(N)},$$

where

1. $\mathbf{A}^{(n)} = \begin{pmatrix} A_1^{(n)} & A_2^{(n)} & \dots & A_{I_n}^{(n)} \end{pmatrix}$ is an orthogonal matrix of order $(I_n \times I_n)$.

2. $\mathcal{G} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ of which the subtensors $\mathcal{G}_{i_n=\alpha}$, attained by fixing the n th index to α , have the properties of

(i) ordering:

$$\|\mathcal{G}_{i_n=1}\|_F \geq \|\mathcal{G}_{i_n=2}\|_F \geq \dots \geq \|\mathcal{G}_{i_n=I_n}\|_F \geq 0, \quad (4.4)$$

for all possible values of n .

(ii) all-orthogonality : For all possible values of n, α and β , two subtensors $\mathcal{G}_{i_n=\alpha}$ and $\mathcal{G}_{i_n=\beta}$ are orthogonal subject to $\alpha \neq \beta$:

$$\langle \mathcal{G}_{i_n=\alpha}, \mathcal{G}_{i_n=\beta} \rangle = 0 \text{ for } \alpha \neq \beta.$$

$$3. \mathcal{G} = \mathcal{X} \times_1 \mathbf{A}^{(1)T} \times_2 \mathbf{A}^{(2)T} \cdots \times_N \mathbf{A}^{(N)T}.$$

The vector $A_i^{(n)}$ is an n -mode singular vector and the Frobenius-norms $\|\mathcal{G}_{i_n=i}\|_F$, symbolized by $\sigma_i^{(n)}$, are n -mode singular values of \mathcal{X} .

Theorem 4.3.1.2. (i) Let r_n be equal to the highest index for which $\|\mathcal{G}_{i_n=r_n}\|_F > 0$ in (4.4). We have

$$R_n = \text{rank}_n(\mathcal{X}) = r_n.$$

$$(ii) \|\mathcal{X}\|_F^2 = \sum_{i=1}^{R_1} (\sigma_i^{(1)})^2 = \cdots = \sum_{i=1}^{R_N} (\sigma_i^{(N)})^2 = \|\mathcal{G}\|_F^2.$$

(iii) By discarding the lowest n -mode singular values $\sigma_{I'_n+1}^{(n)}, \sigma_{I'_n+2}^{(n)}, \dots, \sigma_{R_n}^{(n)}$ for given values of $I'_n, 1 \leq n \leq N$, define a tensor \mathcal{Y} , i.e., set the corresponding parts of \mathcal{G} equal to zero. Then we have

$$\|\mathcal{X} - \mathcal{Y}\|_F^2 \leq \sum_{i_1=I'_1+1}^{R_1} (\sigma_{i_1}^{(1)})^2 + \sum_{i_2=I'_2+1}^{R_2} (\sigma_{i_2}^{(2)})^2 + \cdots + \sum_{i_N=I'_N+1}^{R_N} (\sigma_{i_N}^{(N)})^2.$$

4.3.2 CANDECAMP/PARAFAC (CP) Decomposition

A CP decomposition of $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is given by [52],

$$\mathcal{X} = \mathcal{I} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)} \equiv [[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \dots, \mathbf{A}^{(N)}]], \quad (4.5)$$

where is the identity tensor \mathcal{I} has ones along the superdiagonal and zeros elsewhere, $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}, n = 1, \dots, N$, and $R \in \mathbb{N}$.

According to [53], (4.5) can be written element-wise as

$$\mathcal{X}(i_1, i_2, \dots, i_N) = \sum_{r=1}^R \mathbf{A}^{(1)}(i_1, r) \mathbf{A}^{(2)}(i_2, r) \cdots \mathbf{A}^{(N)}(i_N, r).$$

Moreover, another way of rewriting (4.5) is via the outer product

$$\mathcal{X} = \sum_{r=1}^R \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)}.$$

4.3.3 Tensor Train (TT) Decomposition

A TT decomposition of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is of the form [59],

$$\mathcal{X}(i_1, \dots, i_N) = \mathcal{G}_1(i_1) \cdot \dots \cdot \mathcal{G}_N(i_N).$$

The three way tensors $\mathcal{G}_n \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$ are called TT cores. "Boundary conditions" $r_0 = r_N = 1$ are imposed, and TT ranks are defined as (r_1, \dots, r_{N-1}) .

In the index form, the decomposition is written as

$$\mathcal{X}(i_1, i_2, \dots, i_{N-1}, i_N) = \sum_{k_1, k_2, \dots, k_{N-2}, k_{N-1}} \mathcal{G}_1(i_1, k_1) \mathcal{G}_2(k_1, i_2, k_2) \cdots \mathcal{G}_N(k_{N-1}, i_N). \quad (4.6)$$

The pattern of (4.6) can be conceptualized as a "train," where each core is represented as a "carriage." The initial and terminal carriages are matrices, while those in between are order-3 tensors. Adjacent carriages are linked by a singular contraction.

4.4 Transient Matrix in Kronecker Product Form

The CME matrix \mathcal{M} is sparse under any order relation imposed on the state space, and it is natural to use the lexicographic order on the state space \mathbf{X} as a subset of \mathbb{N} , the set of nonnegative natural numbers. If we impose the lexicographic order on a state space \mathbf{X} in the form of a window of lower bound $(lb^{(1)}, \dots, lb^{(N)})$ and upper bound $(ub^{(1)}, \dots, ub^{(N)})$, i.e., \mathbf{X} consists of states $\mathbf{x} = (x_1, \dots, x_N)^T$ such that $lb^{(s)} \leq x_s \leq ub^{(s)}, \forall s = 1, \dots, N$, then the transition rate matrix can be decomposed into a sum of Kronecker products [40, 49]

$$\mathcal{M} = \sum_{k=1}^M \left(\bigotimes_{s=1}^N \mathbf{S}_k^{(s)} - \mathbf{I} \right) \mathbf{M}_k. \quad (4.7)$$

Each term in the sum corresponds to a chemical reaction; $\mathbf{S}_k^{(s)}$ is a ‘shift-diagonal’ matrix corresponding to the change in species s when reaction k happens; \mathbf{I} is the identity matrix; and \mathbf{M}_k is the diagonal matrix that stores the values of the propensity function a_j . Precisely, let $I_s = ub^{(s)} - lb^{(s)} + 1$ be the window size for species S , $n = I_1 \cdot \dots \cdot I_N$ be the total number of states in \mathbf{X} , and denote $\nu_k = \left(\nu_k^{(1)}, \dots, \nu_k^{(N)}\right)^T$. The identity matrix \mathbf{I} is then of order n , $\mathbf{S}_k^{(s)} \in \mathbb{R}^{I_s \times I_s}$ is given by

$$\mathbf{S}_k^{(s)} = \begin{cases} \begin{pmatrix} 0 & \cdots & 1 & & \\ & \ddots & & \ddots & \\ & & \ddots & & 1 \\ & & & \ddots & \vdots \\ & & & & 0 \end{pmatrix} & \text{shifted up } \left| \nu_k^{(s)} \right| \text{ rows if } \nu_k^{(s)} < 0, \\ \begin{pmatrix} 0 & & & & \\ \vdots & \ddots & & & \\ 1 & & \ddots & & \\ & \ddots & & \ddots & \\ & & 1 & \cdots & 0 \end{pmatrix} & \text{shifted down } \nu_k^{(s)} \text{ rows if } \nu_k^{(s)} \geq 0, \end{cases}$$

and

$$\mathbf{M}_k = (\alpha_k(\mathbf{x}_1), \alpha_k(\mathbf{x}_2), \dots, \alpha_k(\mathbf{x}_n)), \quad (4.8)$$

where $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ are states of \mathbf{X} in the increasing lexicographic order. Furthermore, by assuming that the propensity functions are separable, i.e.,

$$\alpha_k(\mathbf{x}) = \alpha_k^{(1)}(x_1) \cdot \dots \cdot \alpha_k^{(N)}(x_N), \quad k = 1, \dots, M,$$

where $\alpha_k^{(\cdot)} \geq 0$ are scalar functions, (4.8) can be rewritten as

$$\mathbf{M}_k = \otimes_{s=1}^N \alpha_k^{(s)}, \quad (4.9)$$

where

$$\alpha_k^{(s)} = \left(\alpha_k^{(s)}(lb^{(s)}), \alpha_k^{(s)}(lb^{(s)} + 1), \dots, \alpha_k^{(s)}(ub^{(s)}) \right), \quad s = 1, \dots, N.$$

The lower bounds in [72] is $lb^{(s)} = 0$ and $ub^{(s)} = I_s - 1$. Therefore,

$$\alpha_k^{(s)} = \left(\alpha_k^{(s)}(0), \alpha_k^{(s)}(1), \dots, \alpha_k^{(s)}(I_s - 1) \right), \quad s = 1, \dots, N.$$

Under these assumptions, the transition matrix can be represented by simple matrices of small sizes,

$$\mathcal{M} = \sum_{k=1}^M \left(\otimes_{s=1}^N \mathbf{S}_k^{(s)} - \mathbf{I} \right) \left(\otimes_{s=1}^N \alpha_k^{(s)} \right). \quad (4.10)$$

A central aspect of our research is that both the transition operator \mathcal{M} and the probability distribution \mathbf{P} can be kept as multidimensional arrays (or tensors) that allow us to use tensor decompositions to resolve storage issues.

4.5 Study Findings using a Metapopulation Model

We assume that initially, there are 2 healthy individuals and 1 infected individuals in area 1. In area 2 there are 1 healthy and 2 infected individuals. The initial state vector is $[2, 1, 1, 2]$, and we set the reaction parameters $c_1 = .30$ and $c_2 = 1.0$, and the diffusion parameters $D = 1.0$ and $l = 10$. The RDME gives us all the possible states we can get when those individuals react and diffuse randomly. When we analyze this model without

considering areas and diffusion, the stoichiometric matrix is given by,

$$S = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 2 \end{pmatrix}.$$

The first and second row of the matrix stand for the reactions (3.6) and (3.7) respectively. On the other hand, when we consider the scheme in two areas where diffusion also involved, the stoichiometric matrix becomes,

$$S = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 2 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

In this matrix the first two rows indicate the reactions (3.6) and (3.7) inside area 1. Third and fourth rows are responsible for the same reactions respectively in area 2. Diffusions are characterized by the last four rows. Starting with such a simple initial state $[2, 1, 1, 2]$, we get total 84 possible states. We solve the RDME using the FSP equation (3.5) by truncating the state space to 70, to approximate the probability of the states. The marginal probability of each person is of special relevance to us. Assuming $t = 10$, we approximate the marginal probability distribution of the healthy and infected individuals in two areas. The results are shown in figure 4.4. We use tensors to the metapopulation model. The main concept is to express a high-dimensional tensor as an ordered product of smaller tensors. The tensor train is the name given to this series of smaller tensors. The sequence's tensors are all fixed in size and are joined to one another

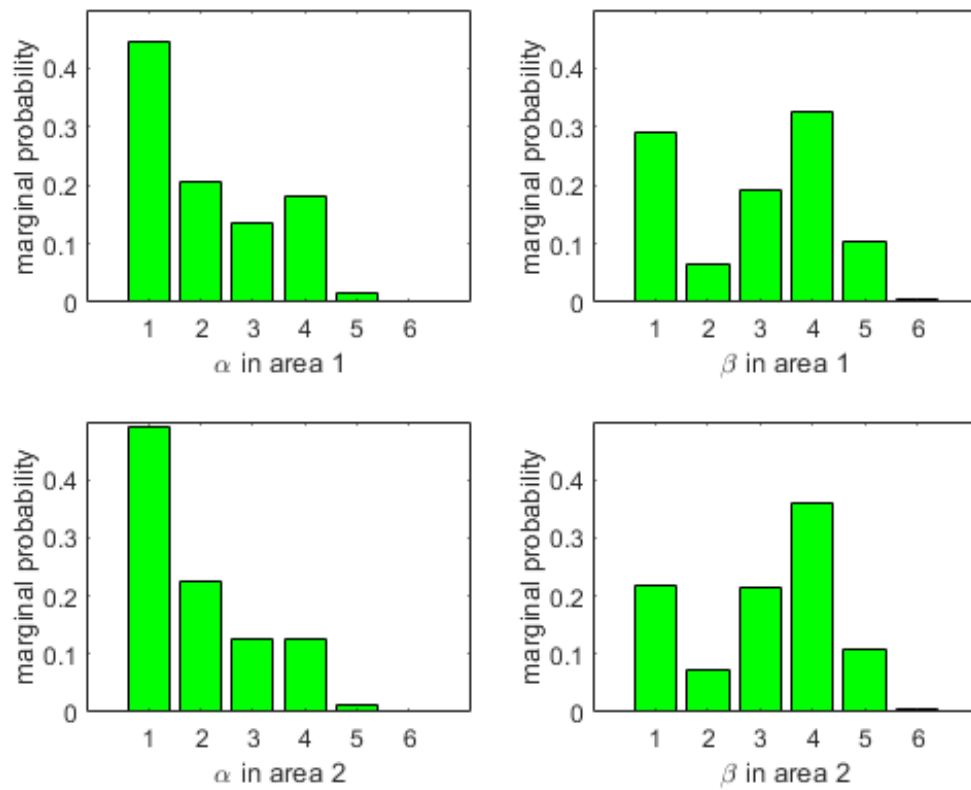


Figure 4.4: Marginal probability distribution of the number of healthy and infected individuals in two areas of a metapopulation model using FSP at time $t = 10$.

by a set of indices. The marginal probability distribution of the individuals are found by using the code in [21]. Figure 4.5 shows the marginal probability distribution of the healthy and infected individuals in two areas using the tensor technique. Figure 4.6 illustrates the accuracy of the implementation of tensors by showing the near alignment of the marginal distributions obtained using two alternative approaches.

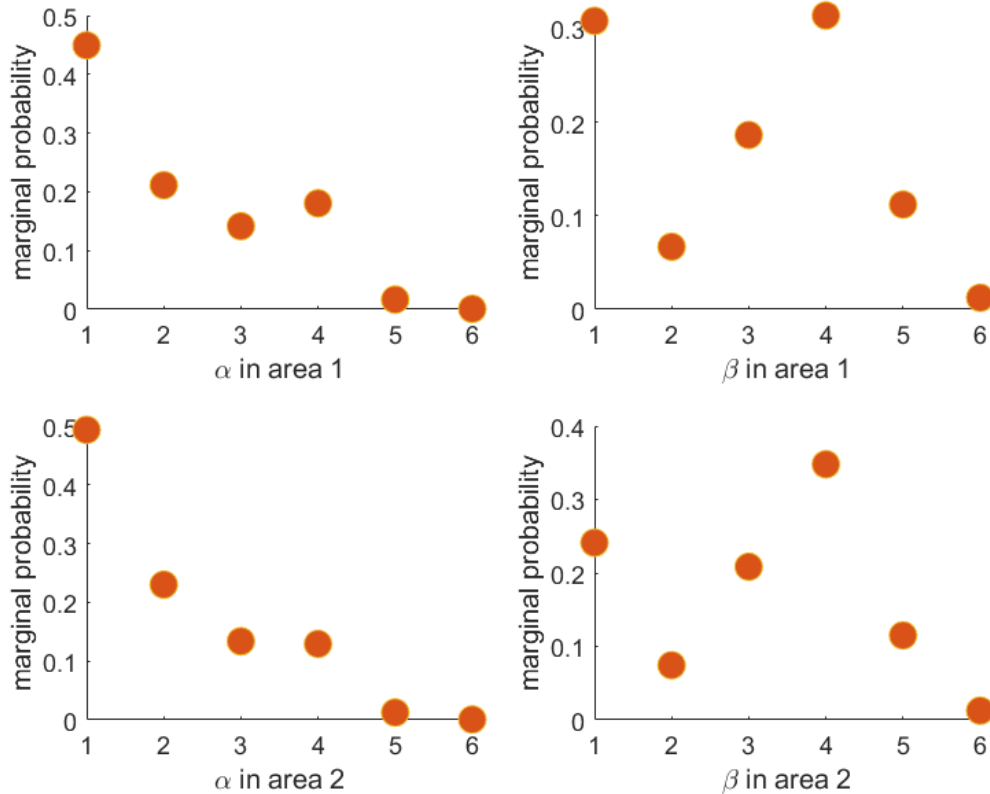


Figure 4.5: Marginal probability distribution of the number of healthy and infected individuals in two areas of a metapopulation model using tensors at time $t = 10$.

4.6 Discussion and Conclusion

We have applied the RDME formulation in a metapopulation model that includes both reaction and diffusion processes and we have utilized FSP and tensors to obtain the marginal probability distribution. Rather than taking into account all possible states, the FSP operates with a reduced state space. On the other hand, we convert the transition

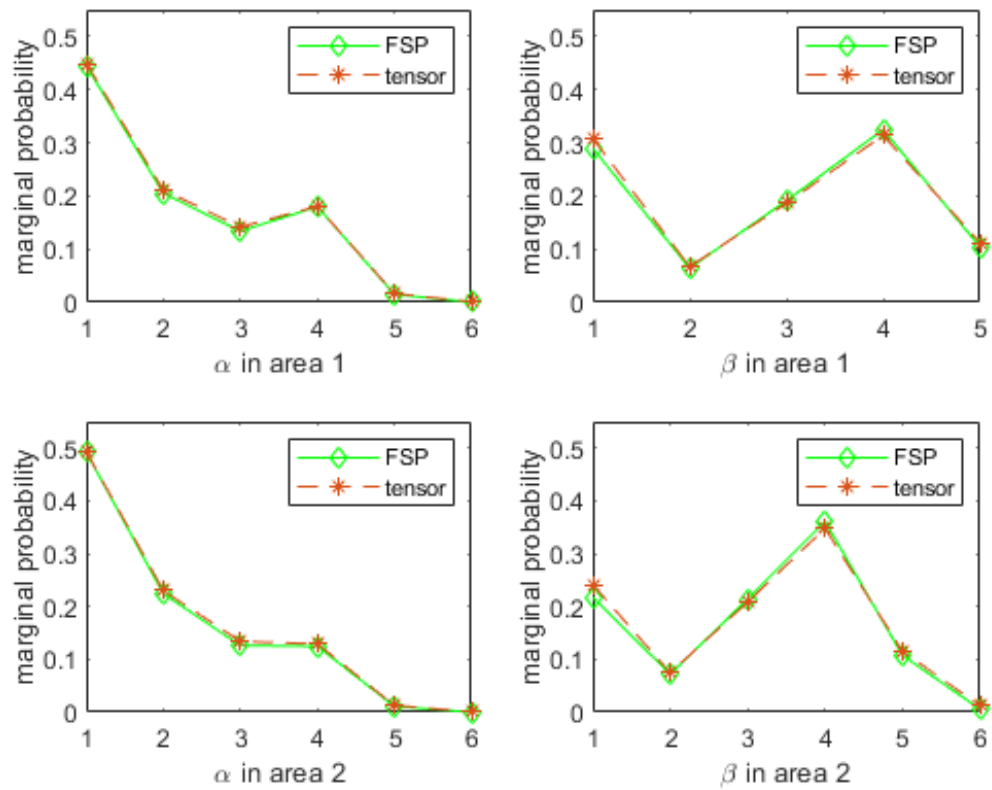


Figure 4.6: Comparison of marginal probability distribution of the number of healthy and infected individuals in two areas of a metapopulation model using FSP and tensors.

matrix into sequence of tensors and that motivates us to use tensor decomposition. Past research has shown that tensors scale very well to tackle multidimensional CME problems, suggesting promising prospects for their application in RDME as we demonstrated in this case study. While the computational performance of the tensor and FSP methods is comparable, the tensor approach offers distinct advantages. It excels in managing high-dimensional problems by efficiently representing extensive state spaces with minimized memory requirements, making it particularly effective in scenarios where FSP becomes computationally expensive or infeasible. Additional benefits include its compatibility with modern numerical and parallel computing frameworks. Its potential extends to systems involving interactions among multiple variables, such as systems biology, chemical reaction networks, and multi-agent systems, highlighting its significance for future research endeavors.

CHAPTER 5 PARALLEL IMPLEMENTATION OF THE STOCHASTIC REACTION-DIFFUSION MASTER EQUATION

This chapter is based on work accepted in Springer book, BICOB-2025. In this chapter, we utilize the RDME framework in several biological models, integrating reaction and diffusion processes, and simulate stochastic trajectories via the SSA algorithm. Due to the substantial computational expense of executing the SSA within the RDME framework, we utilize parallel computing to minimize simulation duration. This chapter delves into the principles and practices of parallelism in RDME models, beginning with its definition and key types, followed by an examination of the role of high-performance computing resources like the Alabama Supercomputer. Practical methods for implementing parallelism and an analysis of its impact on computational efficiency is also explored, highlighting how parallelism continues to shape the future of scientific discovery. These findings highlight the significance of parallel computing in enhancing large-scale biological simulations, prompting additional investigation into sophisticated computational methodologies.

5.1 History of Parallel Computing

The conceptual foundation of parallelism can be traced back to the 1950s and 1960s, when pioneering computer systems like the IBM Stretch and Burroughs D825 introduced early forms of multiprocessing. These systems demonstrated the potential of using multiple processors to share workloads, paving the way for more advanced parallel architectures. Around the same time, Seymour Cray's work on vector computing, particularly in the CDC 6600, introduced the concept of executing multiple operations simultaneously on a single processor, a technique that became instrumental in scientific computing.

By the 1980s, parallel computing entered a new era with the development of massively

parallel processors (MPP) and supercomputers like the Connection Machine and Cray-2. These systems integrated thousands of processors and demonstrated the scalability of parallelism for solving large-scale problems, including weather simulations, molecular dynamics, and fluid dynamics. During this period, distributed computing also began to emerge, with frameworks designed to harness the power of multiple machines working in concert. Task parallelism, where different tasks are distributed across processors, and data parallelism, where the same operation is applied to different pieces of data, became fundamental paradigms.

The 1990s and 2000s saw parallel computing move beyond high-performance systems into more accessible hardware with the rise of multicore processors and general-purpose GPUs (graphics processing units). These innovations brought parallelism into everyday computing, enabling researchers, engineers, and developers to leverage parallel architectures for a wider range of applications. At the same time, software frameworks like MPI (Message Passing Interface) and OpenMP (Open Multi-Processing) standardized the implementation of parallelism across platforms, making it easier to develop scalable solutions.

5.2 What is Parallel Computing

Parallel computing is a computational paradigm that allows multiple tasks to be executed simultaneously, leveraging the capabilities of modern processors to achieve significant improvements in speed and efficiency. Unlike sequential computing, where tasks are performed one after another on a single processor, parallel computing divides a problem into smaller tasks and assigns them to different processors or computational units to execute concurrently.

The primary goal of parallel computing is to reduce the time required to solve complex problems by taking advantage of the inherent parallelism in many real-world tasks. For example, in SSA, large number of simulations can be computed independently before

combining the results. Similarly, in image processing, each pixel or a block of pixels can be processed in parallel to enhance speed. Parallel computing relies on three main concepts:

Decomposition Breaking down a large problem into smaller, independent tasks that can be solved concurrently.

Coordination Ensuring proper synchronization and communication between tasks, especially when they depend on shared data.

Scalability Ensuring that the performance gains from parallelization increase as more processors or computational units are added.

Modern parallel computing systems are built around hardware architectures that support concurrent execution, such as multi-core CPUs, GPUs, and distributed computing clusters. Software frameworks and programming models, like OpenMP for shared memory systems and MPI for distributed systems, provide the tools to implement parallel algorithms effectively.

5.3 Types of Parallel Computing

Parallel computing encompasses several types, each tailored to specific hardware architectures and problem requirements. These types differ in how tasks are distributed and executed across computational units, and their selection depends on the problem's nature, system resources, and desired performance outcomes. The primary types of parallel computing are:

Task Parallelism Task parallelism involves distributing different tasks or functions across multiple processors, with each processor executing a unique task independently. This type is suitable for problems where tasks are heterogeneous and loosely coupled, such as rendering different parts of a scene in computer graphics or processing different stages of a pipeline in data processing.

Data Parallelism In data parallelism, the same operation is applied to different chunks of data simultaneously. For example, in matrix multiplication or image filtering, each

processor handles a subset of the data, performing identical computations in parallel. Data parallelism is widely used in GPU computing, where thousands of cores process large datasets concurrently.

Shared Memory Parallelism Shared memory parallelism uses a single memory space accessible by all processors. Processors communicate by reading and writing to shared variables. Frameworks like OpenMP are commonly used to implement this type of parallelism on multi-core CPUs. It is suitable for small to medium-sized problems where memory access contention is manageable.

Distributed Memory Parallelism In distributed memory parallelism, each processor has its own private memory, and communication between processors occurs via message-passing protocols like MPI. This type is highly scalable and is used in large computing clusters for simulations, weather modeling, and other high-performance computing applications.

Hybrid Parallelism Hybrid parallelism combines multiple types of parallelism, such as using MPI for distributed computing and OpenMP for shared memory within each node. This approach maximizes the utilization of modern supercomputers that include multi-core CPUs and GPUs.

In this chapter, We will concentrate on shared memory, which will be implemented through the use of OpenMP.

5.4 OpenMP

OpenMP, which stands for Open Multi-Processing, is a shared-memory parallel computing model that provides compiler directives, library routines, and environment variables for managing parallel execution. The "Open" in OpenMP signifies that the standard is publicly accessible, allowing widespread adoption across various domains, including scientific computing, engineering simulations, and high-performance computing applications. Parallelism in OpenMP is categorized into two types: coarse-grained and

fine-grained. In coarse-grained parallelism, the computational domain is divided into multiple subdomains, each assigned to different processors, reducing inter-processor communication overhead but requiring careful load balancing. In contrast, fine-grained parallelism distributes iterations of do-loops across multiple processors, where each processor executes a subset of the iterations, leading to better utilization of available threads for iterative workloads. Our implementation focuses on fine-grained parallelism, ensuring that do-loop iterations are efficiently assigned to different threads, thereby maximizing computational throughput and minimizing idle processor time.

The directive sentinel `!$OMP` is utilized to compile OpenMP programs. The exclamation point `!"` is employed to ensure that a standard compiler recognizes the line as a comment. An OpenMP compiler interprets the line as a directive, activating parallel capabilities. To delineate a parallel zone in OpenMP, we utilize the subsequent directive pair.

```
!$OMP PARALLEL
    write code here
!$OMP END PARALLEL
```

The code segment between two parallel directives executes concurrently, whereas the code sections before and succeeding this parallel region execute sequentially. Each core or thread is assigned a designated set of computational duties. The master thread remains inactive until all other threads in the parallel region have finished executing. The program resumes executing the sequential code following the parallel block only after this synchronization step. This synchronization guarantees that all threads have completed their designated responsibilities prior to the program's continuation. Figure 5.1 depicts a serial application partitioned into three threads that execute concurrently.

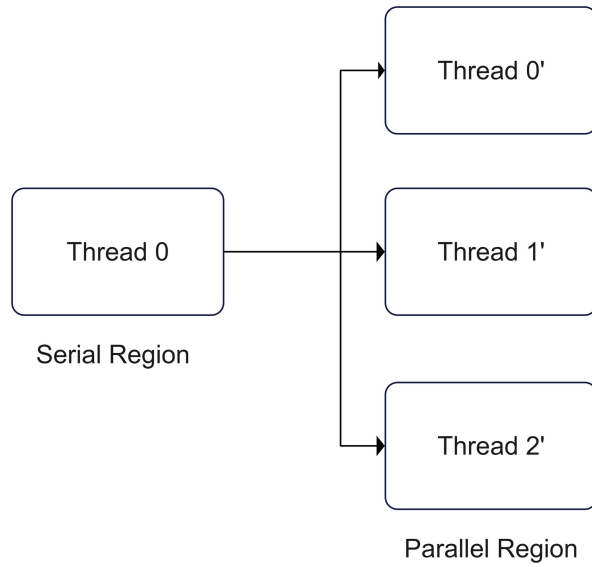


Figure 5.1: Parallel demonstration.

Parallel regions can be nested as well, as demonstrated by the following example of a nested directive pair.

```
!$OMP PARALLEL
  !$OMP PARALLEL
    write code here
  !$OMP END PARALLEL
!$OMP END PARALLEL
```

The first parallel directive splits the serial code into multiple threads. Likewise, the inner parallel directive further divides its own threads into additional threads. Figure 5.2 demonstrates how a serial program is divided into two levels of nested threads.

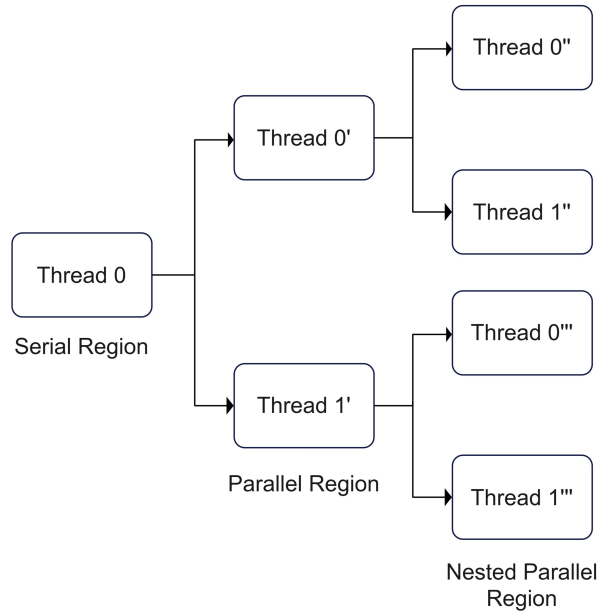


Figure 5.2: Nested parallel demonstration.

Iterations of a do-loop can be assigned to different threads for parallel execution. To achieve this, we define a parallel region around the do-loop using the following directive pair.

```

!$OMP DO
    do i = 1, 2000
    write code here
    end do
!$OMP END DO
  
```

This sample consists of 2000 independent iterations distributed across 4 threads, with each thread executing 500 iterations. Figure 5.3 depicts this process.

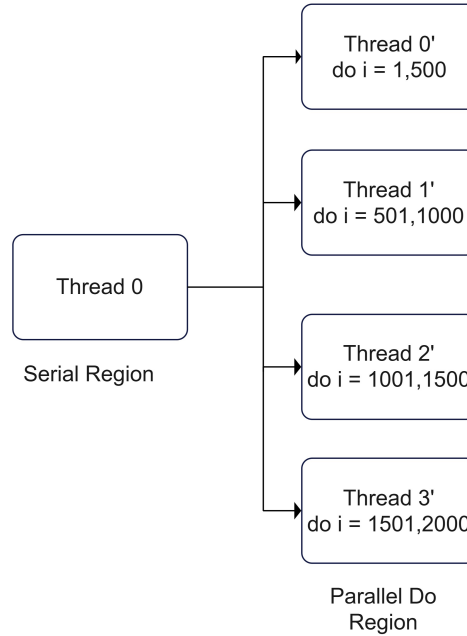


Figure 5.3: Parallel do demonstration.

5.5 Alabama Supercomputer Center (ASC)

5.5.1 About ASC

The Alabama Supercomputer Authority (ASA), established by the Alabama Legislature in 1989, plays a pivotal role in advancing technology-driven scientific research and development across the state. ASA administers and operates the Alabama Supercomputer Center (ASC), providing high-performance computing (HPC) services at no cost to faculty, staff, and enrolled students of public schools, colleges, and universities in Alabama. The ASC supports a wide range of research disciplines, including computational biology, engineering simulations, climate modeling, artificial intelligence, and data-intensive applications, enabling researchers to process complex computations more efficiently. ASC is a high-performance computing cluster, comprising multiple individual computers functioning as a single entity. Each individual computer within the cluster is referred to as a node. The computers within each cluster operate at varying speeds; clusters equipped with newer, faster technology will outperform those with older, slower systems. The ASC

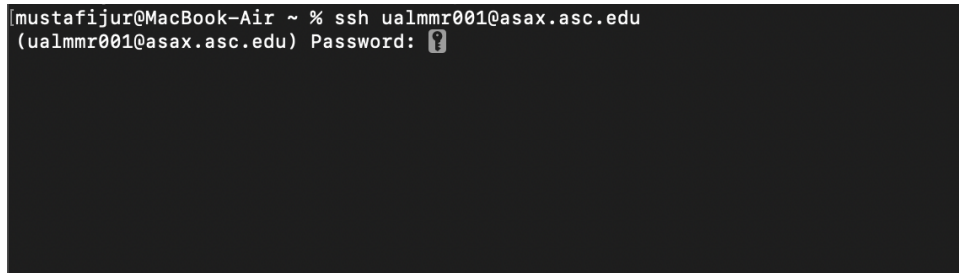
facilitates the rapid execution of extensive programs in parallel. Users have access to numerous core processors and several terabytes of memory for data storage required by programs.

5.5.2 Accessing the ASC

Begin by launching a command Prompt (Windows OS) or terminal (Linux) window. You need to use the “ssh” command to access the ASC. start by entering:

```
ssh username@asax.asc.edu
```

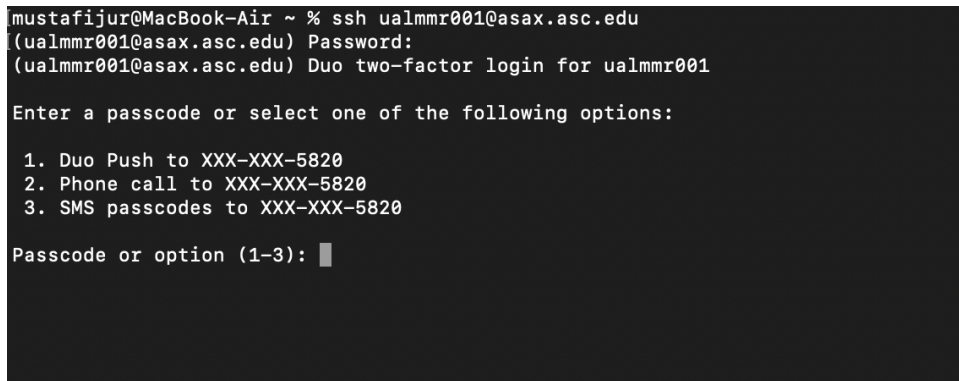
as illustrated in the figure 5.4. The username section must be your personal HPC username. Upon execution, the terminal will request you for your password.



```
mustafijur@MacBook-Air ~ % ssh ualmmr001@asax.asc.edu  
(ualmmr001@asax.asc.edu) Password: ?
```

Figure 5.4: Accessing the ASC using terminal window.

You will subsequently receive a dual-factor authentication prompt shown in figure 5.5. It is advisable to utilize the “DUO Push” approach, as it is the most secure option available.



```
mustafijur@MacBook-Air ~ % ssh ualmmr001@asax.asc.edu  
(ualmmr001@asax.asc.edu) Password:  
(ualmmr001@asax.asc.edu) Duo two-factor login for ualmmr001  
  
Enter a passcode or select one of the following options:  
  
1. Duo Push to XXX-XXX-5820  
2. Phone call to XXX-XXX-5820  
3. SMS passcodes to XXX-XXX-5820  
  
Passcode or option (1-3): █
```

Figure 5.5: Two step authentication before login.

After that you will be greeted by the following banner which indicates that you have successfully logged in and ready to use the ASC.

```
Success. Logging you in...
Last failed login: Sun Feb 16 20:17:00 CST 2025 from 68.62.153.227 on ssh:notty
There was 1 failed login attempt since the last successful login.
-----
The HPC systems are NOT certified for HIPAA, ITAR, 800-171, DoD classified,
GDPR, or most other government-defined security standards.

The HPC documentation website is https://hpcdocs.asc.edu/ If you have any
questions please contact hpc@asc.edu or call the helpdesk at 256-971-7448
or 800-338-8320 if you are outside of Huntsville. We can often give the
fastest assistance if we receive an email with the directory and job number.
-----
ualmnr001@asaxlogin2:~> █
```

Figure 5.6: Information upon successful login.

5.5.3 Necessary Linux Commands

The HPC supercomputer system is a strictly Linux-based system, necessitating proficiency in terminal navigation inside the Linux environment. Below is a list of essential Linux commands for HPC users.

- **ls** - Used to check the contents of the present working directory. The command **ls-a** will additionally display hidden files, while **ls-l** provides detailed information for each file, including permissions, size, and ownership.
- **mkdir** - Used to create a new subdirectory within the present working directory.
- **mv** - Used to move files from one location to another.
- **cp** - Used to copy and paste files from one location to another.
- **rm** - Used to remove / delete files and directories.
- **vi** - A text editor used to create, write, and change various text files including scripts
- **grep** - Used to filter text based on a specific word or phrase.
- **awk** - Used to filter data into rows and columns.
- **echo** - Used to print text to the terminal.
- **tr** - Used to remove unwanted characters from text.

- **qstat** - Used to check the status of active jobs.
- **jobinfo** - Used to see the detailed information about a specific job, such as memory use and parallel efficiency.
- **qdel** - Used to delete queued jobs.

5.5.4 Submitting Job into Queue

Access the directory where your files are stored. Shell script files encompass all the commands required to execute a code. This file must be saved as `openmp.sh`, where `.sh` indicating a shell script file. To execute a script, input the following command.

```
chmod +x openmp.sh
run_script openmp.sh
```

A sequence of prompts will be displayed shown in figure 5.7

```

ualmmr001@asaxlogin2:code> run_script openmp.sh
This runs a script in the current directory via the queue system
This script lets the job use multiple processors on the same node.
Report problems and post questions to the HPC staff (hpc@asc.edu)

Choose a batch job queue:

Queue           Wall Time      Mem  # Cores
-----
express         4:00:00       16gb  1-4
small           60:00:00       4gb   1-8
medium          150:00:00     16gb  1-16
large           360:00:00    120gb  1-128
bigmem          360:00:00   130-500gb  1-32
benchmark       24:00:00    1400gb  1-192

Your job will have a shorter wait time if your memory request is
reasonable (about 20% more than needed), and your time request is
reasonable (about 50% more than needed).
Find this out by running 'jobinfo -j JOB_NUMBER' for a correctly
completed job.

Enter Queue Name (default <cr>: small)

```

Figure 5.7: Selecting desired parallel environment for the job.

you need to choose the following information,

- Queue size (based on the memory, number of cores, time to be used).
- Number of processors (cores/threads).

- Time limit for the job.
- Memory limit for the job.
- Name of the job.
- The cluster of high performance computers you want to use.

Upon selecting the desired features, your task will begin and will be assigned an ID number, enabling you to cancel the task or get additional information regarding it as shown in the figure below.

```

=====
====          Summary of your script job          =====
=====
The script file is: openmp.sh
The time limit is 150:00:00 HH:MM:SS.
The memory limit is: 16gb
The job will start running after: 202502162021.34
Job Name: openmpshSCRIPT
Queue: -q medium
Constraints:
Queue submit command:
qsub -q medium -j oe -N openmpshSCRIPT -a 202502162021.34 -r n -M mrahman24@crim
son.ua.edu -l walltime=150:00:00 -l select=ncpus=16:mpiprocs=16:mem=16000mb

236240.asax-pbs1
ualmmr001@asaxlogin2:code> █

```

Figure 5.8: Submitted job details.

5.6 Parallel Stochastic Simulation Algorithm

The parallel stochastic simulation method is a modification of the SSA. Let TSims be the total number of independent simulations to be executed concurrently. The algorithm is given by the following.

Algorithm 1 Parallel SSA

Initial State: $\mathbf{X}(t_0) = x_0$, Final Time: t_f

$t := t_0$;

!\$OMP PARALLEL q=1, TSims $t < t_f$

$x = \mathbf{X}(t)$

Evaluate $\alpha_j(x)$ and $\alpha_0(x)$;

Generate two random numbers r_1, r_2 from $U(0, 1)$

$j := \min\{j, \sum_{j'=1}^j \alpha_{j'}(x) > r_2 \alpha_0(x)\}$;

$\tau := \ln(1/r_1) / \alpha_0(x)$;

$\tau := \min\{\tau, t_f - t\}$;

$t := t + \tau$;

Check for negative values

$\mathbf{X}(t + \tau) := x + \nu_j$

$\mathbf{X}_q(t_f) = (X_1(t_f), \dots, X_N(t_f))$

!\$OMP END PARALLEL

5.7 Speed-Up and Parallel Efficiency

When measuring the effects of parallel implementation, we focus on two terms; speed-up and parallel efficiency. We use the Amdahl's law [44]:

$$\text{Speed-Up} = \frac{time_1}{time_p} \qquad \text{Parallel Efficiency} = \frac{\left(\frac{time_1}{time_p}\right)}{p}$$

where $time_p$ is the computational time using p cores and $p = 1, 2, 4, 8, 16$.

5.8 Numerical Results

5.8.1 Metapopulation Model

For experimental purpose, we assume that initially, there are 100 healthy individuals and 90 infected individuals in area 1. Similarly, in area 2 there are 90 healthy and 100 infected individuals. The initial state vector is $[100, 90, 90, 100]^T$, and we set the reaction parameters $c_1 = .80$ and $c_2 = .30$, and the diffusion parameters $D = 1.0$ and $l = 10$, resulting $d = D/l^2 = 1.0/100 = .01$.

Table 5.1: Computational time with different number of cores for parallel stochastic simulation of metapopulation model.

1, 000, 000, 000 runs	
1 Core:	42.46 sec
2 Cores:	20.27 sec
4 Cores:	10.29 sec
8 Cores:	5.93 sec
16 Cores:	2.04 sec

We use FORTRAN and OpenMP to implement the SSA algorithm for the metapopulation model employing the RDME. We use a do-loop of 1, 000, 000, 000 iterations. The Alabama Super Computer (ASC), a cluster of high performance supercomputers, is used to simulate the trajectories. Table 5.1 shows the computational time for computing 1, 000, 000, 000 individual simulations of the SSA algorithm and figure 5.9 shows the trend. We are able to analyze the speed-up and the parallel efficiency of the SSA algorithm. Figure 5.10 and 5.11 shows the analysis of the parallel implementation.

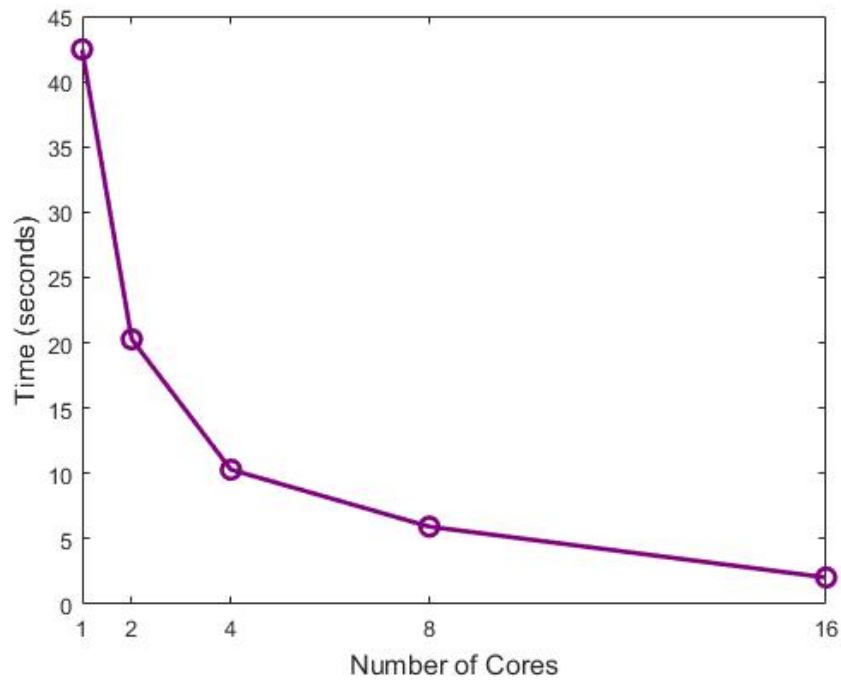


Figure 5.9: Decreasing computational time with different number of cores for parallel stochastic simulation of metapopulation model.

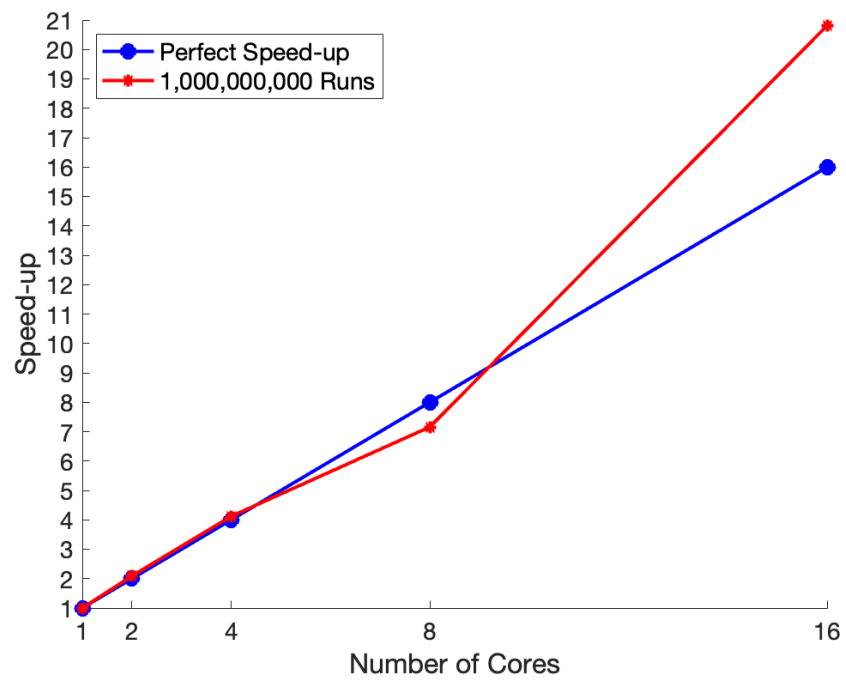


Figure 5.10: Computational speed-up of parallel implementation in metapopulation model.

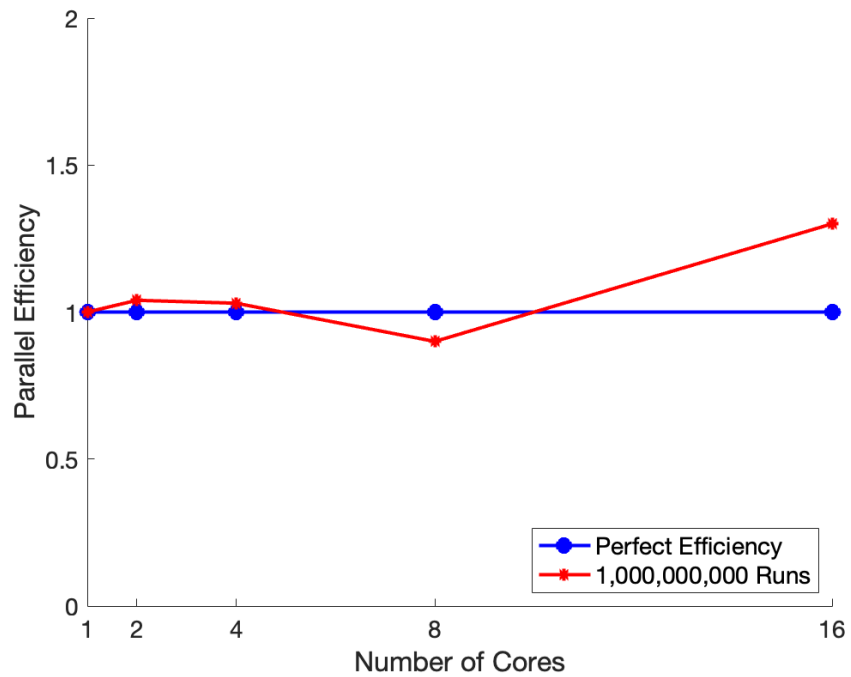


Figure 5.11: Parallel efficiency in metapopulation model.

5.8.2 Birth-Death Process

Consider that initially, there are 100 protein molecules in cell 1, and 100 protein molecules in cell 2. The initial state vector is $[100, 100]^T$, and we set the reaction parameters $c_1 = .60$ and $c_2 = .20$, and the diffusion parameters $D = 1.0$ and $l = 10$, resulting $d = D/l^2 = 1.0/100 = .01$.

Table 5.2: Computational time with different number of cores for parallel stochastic simulation of birth-death model.

1, 000, 000, 000 runs	
1 Core:	39.66 sec
2 Cores:	18.62 sec
4 Cores:	9.04 sec
8 Cores:	4.64 sec
16 Cores:	3.08 sec

We employ parallel processing for the birth-death model using RDME. We use a do-loop of 1, 000, 000, 000 iterations. Table 5.2 shows the computational time for computing 1, 000, 000, 000 individual simulations of the SSA algorithm and figure 5.12 shows the trend. The speed-up and the parallel efficiency of the SSA algorithm are shown in figure 5.13 and 5.14.

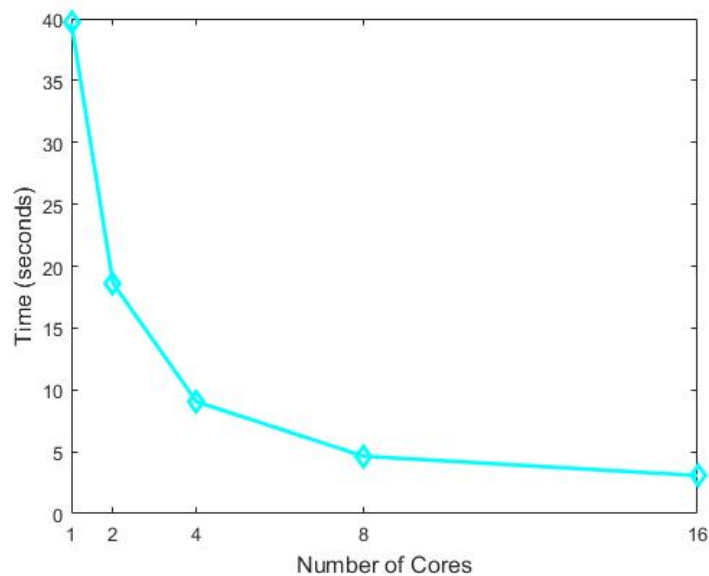


Figure 5.12: Decreasing computational time with different number of cores for parallel stochastic simulation of birth-death model.

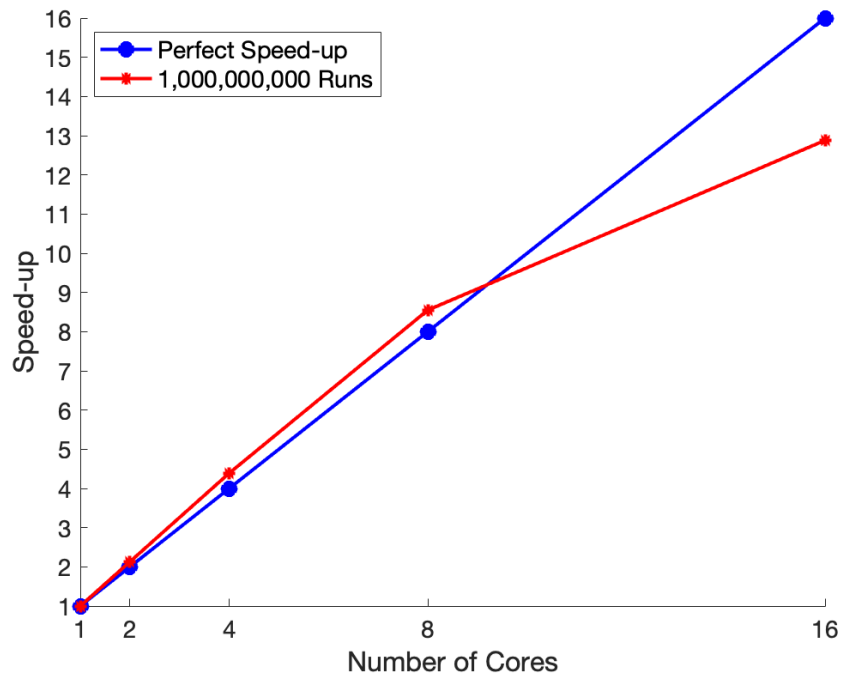


Figure 5.13: Computational speed-up of parallel implementation in birth-death model.

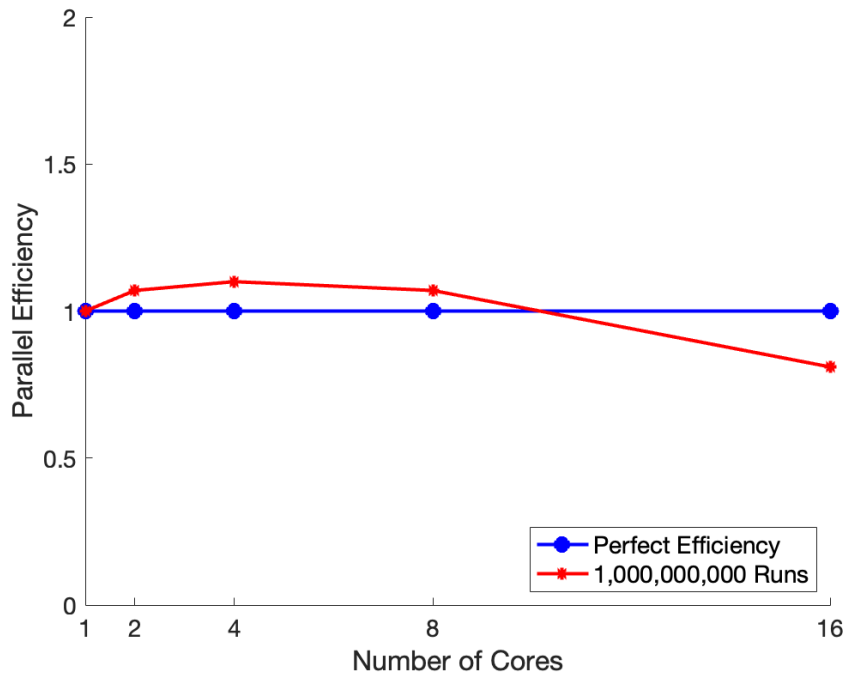


Figure 5.14: Parallel efficiency in birth-death model.

5.8.3 Schögl Model

The Schögl model [71] is a reaction system used in theoretical chemistry and statistical physics to describe bistability in autocatalytic chemical reactions. Named after Friedrich Schögl, this model captures the essential dynamics of a system that can switch between two stable states depending on initial conditions. It consists of a single chemical species undergoing autocatalytic production and depletion through a set of reaction steps. The Schögl model is widely studied in stochastic processes, serving as a fundamental example of noise-driven transitions in chemical and biological systems. The reactions are given by,



We consider two neighboring compartments where the species can jump back and forth. We label the compartments as Compartment 1 and Compartment 2. Thus, the system can be represented as a reaction-diffusion process in which reactions (5.1) through (5.4) occur independently inside both compartments, while molecules can move between Compartment 1 and Compartment 2. We model the scheme with RDME consisting of four reactions with two compartments. Let A_1 be the number of molecules of A in compartment 1, A_2 the number of molecules of A in compartment 2. Similarly, X_1 be the number of molecules of X in compartment 1, X_2 the number of molecules of X in compartment 2. And finally, B_1 be the number of molecules of B in compartment 1, B_2 the number of molecules of B in compartment 2. The reactions R_1 through R_8 in table 5.3 reflect (5.1) through (5.4) inside both compartments, while R_9 through R_{14} represent jump between compartments.

Table 5.3: Reactions, propensities and state change vectors of Schögl model.

Reaction	Propensity	State change vector
$R_1: A_1 + 2X_1 \xrightarrow{c_1} 3X_1$	$c_1 \times A_1 \times X_1 \times (X_1 - 1)$	$[-1, -1, 0, 0, 0, 0]$
$R_2: 3X_1 \xrightarrow{c_2} A_1 + 2X_1$	$c_2 \times X_1 \times (X_1 - 1) \times (X_1 - 2)$	$[1, 1, 0, 0, 0, 0]$
$R_3: B_1 \xrightarrow{c_3} X_1$	$c_3 \times B_1$	$[0, 1, -1, 0, 0, 0]$
$R_4: X_1 \xrightarrow{c_4} B_1$	$c_4 \times X_1$	$[0, -1, 1, 0, 0, 0]$
$R_5: A_2 + 2X_2 \xrightarrow{c_1} 3X_2$	$c_1 \times A_2 \times X_2 \times (X_2 - 1)$	$[0, 0, 0, -1, -1, 0]$
$R_6: 3X_2 \xrightarrow{c_2} A_2 + 2X_2$	$c_2 \times X_2 \times (X_2 - 1) \times (X_2 - 2)$	$[0, 0, 0, 1, 1, 0]$
$R_7: B_2 \xrightarrow{c_3} X_2$	$c_3 \times B_2$	$[0, 0, 0, 0, 1, -1]$
$R_8: X_2 \xrightarrow{c_4} B_2$	$c_4 \times X_2$	$[0, 0, 0, 0, -1, 1]$
$R_9: A_1 \xrightarrow{d} A_2$	$d \times A_1$	$[-1, 0, 0, 1, 0, 0]$
$R_{10}: X_1 \xrightarrow{d} X_2$	$d \times X_1$	$[0, -1, 0, 0, 1, 0]$
$R_{11}: B_1 \xrightarrow{d} B_2$	$d \times B_1$	$[0, 0, -1, 0, 0, 1]$
$R_{12}: A_2 \xrightarrow{d} A_1$	$d \times A_2$	$[1, 0, 0, -1, 0, 0]$
$R_{13}: X_2 \xrightarrow{d} X_1$	$d \times X_2$	$[0, 1, 0, 0, -1, 0]$
$R_{14}: B_2 \xrightarrow{d} B_1$	$d \times B_2$	$[0, 0, 1, 0, 0, -1]$

We assume that initially, there are 100 molecules of A , 50 molecules of X , and 200 molecules of B in compartment 1. Similarly, in compartment 2 there are 80 molecules of A , 40 molecules of X , and 150 molecules of B . The initial state vector is $[100, 50, 200, 80, 40, 150]^T$, and we set the reaction parameters $c_1 = .10$, $c_2 = 2.0$, $c_3 = .10$ and $c_4 = 1.0$, and the diffusion parameters $D = 1.0$ and $l = 10$, resulting $d = D/l^2 = 1.0/100 = .01$. All the state change vectors in table 5.3 form the stoichiometric matrix of the RDME framework.

$$\mathbf{S} = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

We employ parallel processing for the Schögl model using RDME. We use a do-loop of 1,000,000,000 iterations. Table 5.4 shows the computational time for computing 1,000,000,000 individual simulations of the SSA algorithm and figure 5.15 shows the trend. The speed-up and the parallel efficiency of the SSA algorithm are shown in figure 5.16 and 5.17.

Table 5.4: Computational time with different number of cores for parallel stochastic simulation of Schögl model.

1,000,000,000 runs	
1 Core:	74.79 sec
2 Cores:	39.02 sec
4 Cores:	19.64 sec
8 Cores:	13.21 sec
16 Cores:	5.10 sec

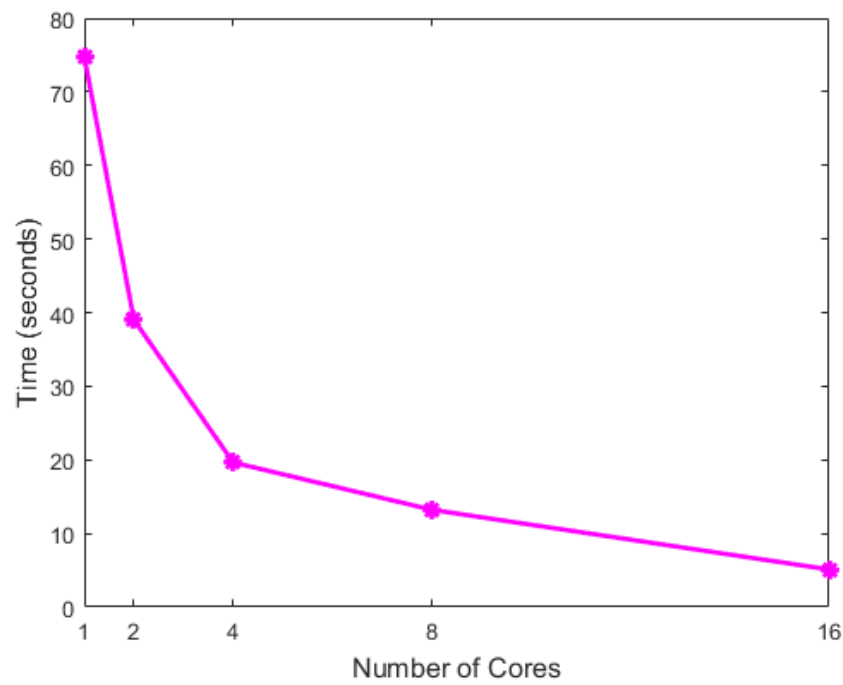


Figure 5.15: Decreasing computational time with different number of cores for parallel stochastic simulation of Schögl model.

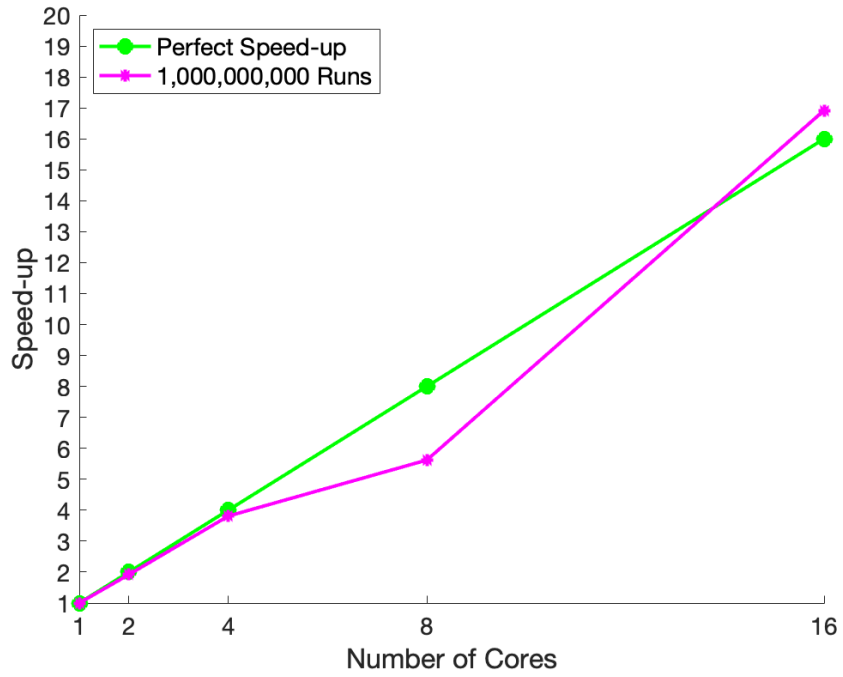


Figure 5.16: Computational speed-up of parallel implementation in Schögl model.

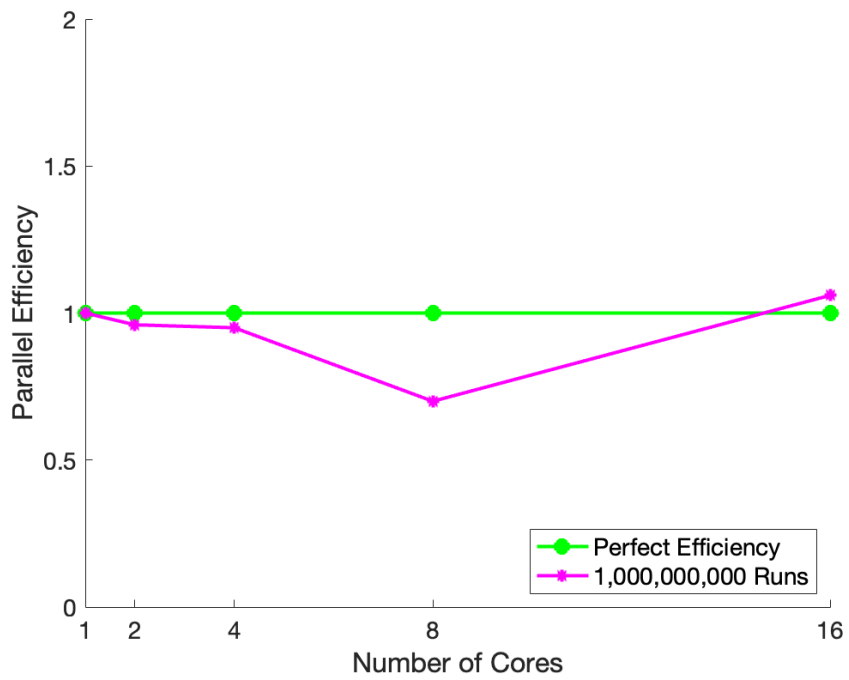


Figure 5.17: Parallel efficiency in Schögl model.

5.8.4 SIR Model

The SIR epidemic model [12, 18, 77] captures the dynamics between three groups: S (Susceptible), I (Infected), and R (Recovered). This model has a wide range of applications, such as modeling the transmission of infectious diseases like influenza, tracking the spread of wildfires, analyzing the spread of infections within cellular systems, and even studying how information or rumors propagate through social networks. A susceptible individual becomes infected after interacting with an infected individual. Over time, infected individuals transition into a recovered individuals. However, recovered individuals can become infected again by first reverting to the susceptible state and then encountering an infected individual. Additionally, individuals in all three states-susceptible, infected, and recovered-can naturally die over time. The reactions of SIR model are given by,



We assume two adjacent areas in which the population can migrate freely. We designate the areas as Area 1 and Area 2. The system can be characterized as a reaction-diffusion process wherein reactions (5.5) to (5.10) can take place independently inside both areas. Also, individuals are permitted to jump between Area 1 and Area 2. We represent the system using a RDME framework comprising six reactions over two areas. Let, S_1 and S_2 represent the number of susceptible individual in Area 1 and Area 2 respectively. $I_1, I_2,$

R_1 , and R_2 have been labeled as a similar way. The first 12 reactions in table 5.5 correspond to equations (5.5) through (5.10) inside both areas, and the last 6 reactions stand for the migration between areas.

Table 5.5: Reactions, propensities, and state change vectors of SIR model.

Reaction	Propensity	State change vector
$S_1 + I_1 \xrightarrow{c_1} 2I_1$	$c_1 \times S_1 \times I_1$	$[-1, -1, 0, 0, 0, 0]$
$I_1 \xrightarrow{c_2} R_1$	$c_2 \times I_1$	$[0, -1, 1, 0, 0, 0]$
$R_1 \xrightarrow{c_3} S_1$	$c_3 \times R_1$	$[1, 0, -1, 0, 0, 0]$
$S_1 \xrightarrow{c_4} \emptyset$	$c_4 \times S_1$	$[-1, 0, 0, 0, 0, 0]$
$I_1 \xrightarrow{c_5} \emptyset$	$c_5 \times I_1$	$[0, -1, 0, 0, 0, 0]$
$R_1 \xrightarrow{c_6} \emptyset$	$c_6 \times R_1$	$[0, 0, -1, 0, 0, 0]$
$S_2 + I_2 \xrightarrow{c_1} 2I_2$	$c_1 \times S_2 \times I_2$	$[0, 0, 0, -1, -1, 0]$
$I_2 \xrightarrow{c_2} R_2$	$c_2 \times I_2$	$[0, 0, 0, 0, -1, 1]$
$R_2 \xrightarrow{c_3} S_2$	$c_3 \times R_2$	$[0, 0, 0, 1, 0, -1]$
$S_2 \xrightarrow{c_4} \emptyset$	$c_4 \times S_2$	$[0, 0, 0, -1, 0, 0]$
$I_2 \xrightarrow{c_5} \emptyset$	$c_5 \times I_2$	$[0, 0, 0, 0, -1, 0]$
$R_2 \xrightarrow{c_6} \emptyset$	$c_6 \times R_2$	$[0, 0, 0, 0, 0, -1]$
$S_1 \xrightarrow{d} S_2$	$d \times S_1$	$[-1, 0, 0, 1, 0, 0]$
$I_1 \xrightarrow{d} I_2$	$d \times I_1$	$[0, -1, 0, 0, 1, 0]$
$R_1 \xrightarrow{d} R_2$	$d \times R_1$	$[0, 0, -1, 0, 0, 1]$
$S_2 \xrightarrow{d} S_1$	$d \times S_2$	$[1, 0, 0, -1, 0, 0]$
$I_2 \xrightarrow{d} I_1$	$d \times I_2$	$[0, 1, 0, 0, -1, 0]$
$R_2 \xrightarrow{d} R_1$	$d \times R_2$	$[0, 0, 1, 0, 0, -1]$

We assume that initially, there are 200 susceptible individuals, 100 infected individuals, and 0 recovered individuals in Area 1. Similarly, in Area 2, there are 190 susceptible, 80 infected, and 0 recovered individuals. The initial state vector is $[200, 100, 0, 190, 80, 0]^T$. We set the reaction parameters $c_1 = .003$, $c_2 = .02$, $c_3 = .007$, $c_4 = .002$, $c_5 = .05$ and $c_6 = .002$, and the diffusion parameters $D = 1.0$ and $l = 10$, resulting $d = D/l^2 = 1.0/100 = .01$.

We employ parallel processing for the SIR model using RDME. We use a do-loop of 1,000,000,000 iterations. Table 5.6 shows the computational time for computing

1,000,000,000 individual simulations of the SSA algorithm and figure 5.18 shows the trend. The speed-up and the parallel efficiency of the SSA algorithm are shown in figure 5.19 and 5.20.

Table 5.6: Computational time with different number of cores for parallel stochastic simulation of SIR model.

1,000,000,000 runs	
1 Core:	87.24 sec
2 Cores:	40.02 sec
4 Cores:	17.84 sec
8 Cores:	11.46 sec
16 Cores:	6.27 sec

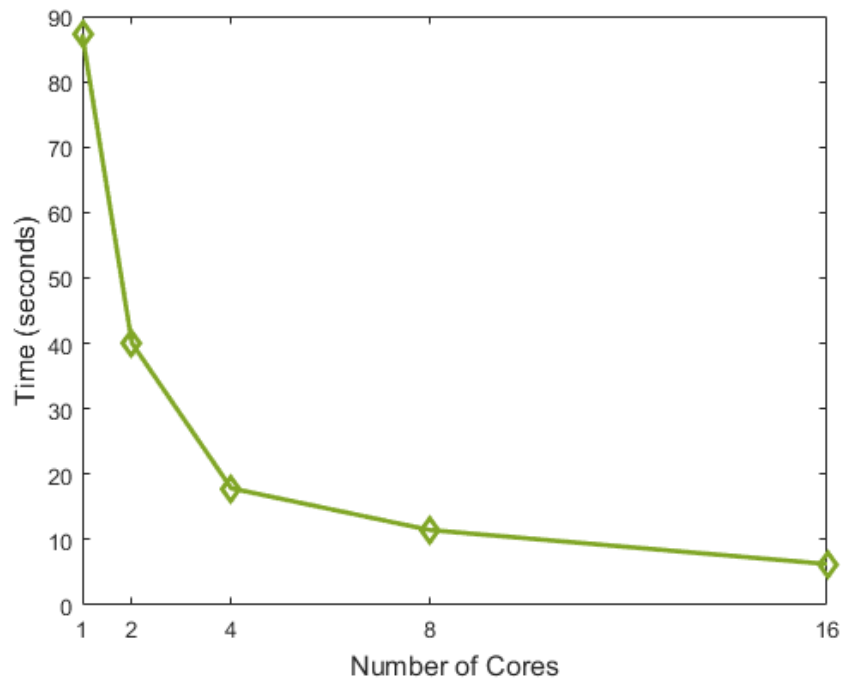


Figure 5.18: Decreasing computational time with different number of cores for parallel stochastic simulation of SIR model.

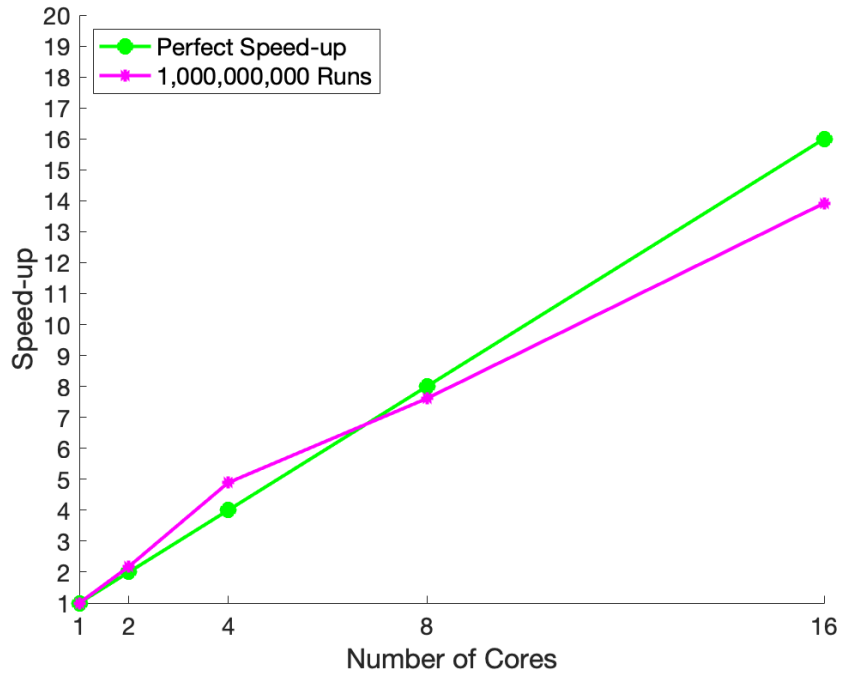


Figure 5.19: Computational speed-up of parallel implementation in SIR model.

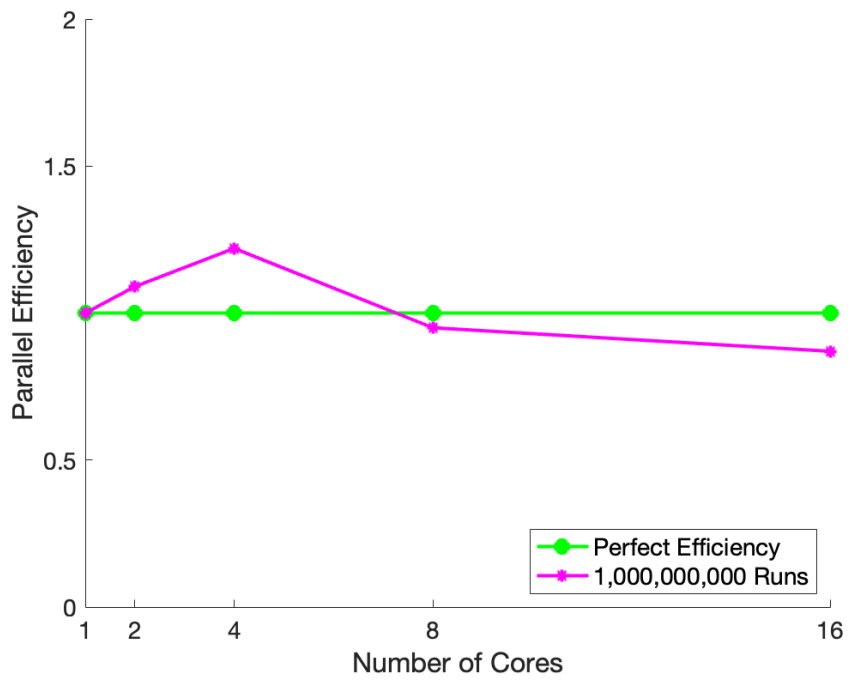


Figure 5.20: Parallel efficiency in SIR model.

5.9 Discussion and Conclusion

We have employed the RDME formulation in several systems biology models that encompass both reaction and diffusion processes. We used the SSA algorithm to simulate stochastic trajectories. Implementing the SSA in the RDME is computationally expensive. We use parallel processing to reduce the computational time of simulating the biological systems. Tables 5.1, 5.2, 5.4, and 5.6 demonstrate that computing time reduces with the utilization of additional cores when working with different biological models. In figure 5.10, the speed-up for the metapopulation model is superlinear with 2, 4, and 16 cores, but sublinear with 8 cores. This signifies that the efficiency exceeds 1 for 2, 4, and 16 cores, while it drops below 1 for 8 cores. The observed superlinear speed-up indicates the number of cores to enhance computing efficiency. Similarly, figure 5.13 depicts the speed-up for the birth-death model. Superlinear performance is attained with 2, 4, and 8 cores, demonstrating efficiencies greater than 1, but shifts to sublinear speed-up with 16 cores, indicating a decline in efficiency below 1. The Schögl model's speed-up is superlinear with 16 cores, as illustrated in figure 5.16, but sublinear with 2, 4, and 8 cores. This indicates that the efficacy of 16 cores exceeds 1, while it falls below 1 for 2, 4, and 8 cores. And lastly, the speed-up for the SIR model is superlinear with 2 and 4 cores, but sublinear with 8 and 16 cores shown in figure 5.19. These findings underscore the significance of parallel computing in addressing the computational requirements of systems biology models that incorporate stochastic reaction-diffusion processes. Future study will focus on using parallel computing approaches to more intricate biological systems. The emphasis will be on enhancing efficiency through advanced parallelization strategies or hybrid computing approaches to better handle large-scale biological simulations.

CHAPTER 6 CONCLUSION

In this dissertation, we have integrated the Reaction-Diffusion Master Equation (RDME) to some biological models. We focus on developing and analyzing computational techniques for solving the RDME efficiently, addressing the challenges posed by “the curse of dimensionality” and the run-time associated with stochastic simulations. By leveraging the Finite State Projection (FSP) method, the Stochastic Simulation Algorithm (SSA), and tensor-based approaches, we have proposed strategies to mitigate these computational challenges and enhance the scalability of RDME applications.

In chapter 3 which is devoted to a study of metapopulation model, we employed the RDME formulation to capture both reaction and jump processes. Using FSP and SSA, we determined the marginal probability distribution, with FSP reducing the state space by truncation and SSA introducing statistical noise at low copy numbers. Despite their effectiveness, these methods suffer from high computational costs, particularly as the system’s dimensionality increases. To address this, we proposed the use of tensor train decomposition techniques. By decomposing high-dimensional tensors into a sequence of smaller tensors, we can significantly reduce computational time while maintaining accuracy. This approach has the potential to provide efficient analysis of large-scale systems, with applications extending to ecology, biology, and epidemiology.

Building upon our initial findings, we extended the RDME formulation using tensors. Our results indicate that tensors scale well for tackling multidimensional RDME problems. Compared to FSP, the tensor approach demonstrates advantages in managing high-dimensional problems by efficiently representing extensive state spaces with reduced memory requirements. Additionally, its compatibility with modern numerical and parallel

computing frameworks positions it as a promising tool for future research in systems biology and chemical reaction networks. These findings underscore the significance of tensor-based methods in advancing stochastic modeling methodologies.

Lastly, we explored the computational efficiency of RDME simulations in systems biology models by employing SSA alongside parallel computing strategies. Given the high computational demands of SSA in RDME, we implemented parallel processing to accelerate stochastic trajectory simulations. Our results demonstrate that computational efficiency improves with additional processing cores, with superlinear speed-ups observed at certain configurations due to factors such as improved cache efficiency and reduced memory contention. However, as the number of cores increases, synchronization overhead and memory bandwidth saturation lead to sublinear speed-up, emphasizing the need for optimized parallelization strategies. These insights highlight the importance of parallel computing in addressing the computational requirements of large-scale biological simulations.

Overall, this dissertation contributes to the advancement of stochastic reaction-diffusion modeling by proposing innovative computational techniques that improve the efficiency and scalability of RDME applications. By integrating tensor decomposition, state space reduction methods, and parallel computing strategies, we have laid the foundation for future research aimed at optimizing RDME computations. Future work will focus on further refining parallelization techniques, potentially integrating hybrid computing approaches to enhance the efficiency of simulating complex biological systems. In addition, to predict the state space, a potential use of machine learning algorithms will be another research direction.

REFERENCES

- [1] Marco Ajelli, Bruno Gonçalves, Duygu Balcan, Vittoria Colizza, Hao Hu, José J Ramasco, Stefano Merler, and Alessandro Vespignani. Comparing large-scale computational approaches to epidemic modeling: agent-based versus structured metapopulation models. *BMC infectious diseases*, 10:1–13, 2010.
- [2] R. M. Anderson, R. M. May, and B. Anderson. *Infectious Diseases of Humans: Dynamics and Control*. Oxford University Press, Oxford, 1992.
- [3] Steven S Andrews and Dennis Bray. Stochastic simulation of chemical reactions with spatial resolution and single molecule detail. *Physical biology*, 1(3):137, 2004.
- [4] Andrea Apolloni, Chiara Poletto, José J Ramasco, Pablo Jensen, and Vittoria Colizza. Metapopulation epidemic models with heterogeneous mixing and travel behaviour. *Theoretical Biology and Medical Modelling*, 11:1–26, 2014.
- [5] Florence Baras and M Malek Mansour. Reaction-diffusion master equation: A comparison with microscopic simulations. *Physical Review E*, 54(6):6139, 1996.
- [6] Andrea Baronchelli, Michele Catanzaro, and Romualdo Pastor-Satorras. Bosonic reaction-diffusion processes on scale-free networks. *Physical Review E*, 78(1):016111, 2008.
- [7] Howard C Berg. *Random walks in biology*. Princeton University Press, 1993.
- [8] David Bernstein. Simulating mesoscopic reaction-diffusion systems using the gillespie algorithm. *Physical Review E*, 71(4):041103, 2005.
- [9] CR Biologies. Théorie générale de l’action de quelques diastases par victor henri [cr acad. sci. paris 135 (1902) 916–919]. *CR Acad. Sci. Paris*, 135:916–919, 1902.
- [10] Adrian J Brown. Xxxvi.—enzyme action. *Journal of the chemical society, transactions*, 81:373–388, 1902.
- [11] Yang Cao, Daniel T. Gillespie, and Linda R. Petzold. Efficient step size selection for the tau-leaping simulation method. *Journal of Chemical Physics*, 124, 2006.
- [12] Settapat Chinviriyasit and Wirawan Chinviriyasit. Numerical modelling of an sir epidemic model with diffusion. *Applied Mathematics and Computation*, 216(2):395–409, 2010.

- [13] Vittoria Colizza, Romualdo Pastor-Satorras, and Alessandro Vespignani. Reaction–diffusion processes and metapopulation models in heterogeneous networks. *Nature Physics*, 3(4):276–282, 2007.
- [14] Vittoria Colizza and Alessandro Vespignani. Epidemic modeling in metapopulation systems with heterogeneous coupling pattern: Theory and simulations. *Journal of theoretical biology*, 251(3):450–467, 2008.
- [15] Athel Cornish-Bowden. The origins of enzyme kinetics. *FEBS letters*, 587(17):2725–2730, 2013.
- [16] Athel Cornish-Bowden. One hundred years of michaelis–menten kinetics. *Perspectives in Science*, 4:3–9, 2015.
- [17] Forrest W Crawford, Lam Si Tung Ho, and Marc A Suchard. Computational methods for birth-death processes. *Wiley Interdisciplinary Reviews: Computational Statistics*, 10(2):e1423, 2018.
- [18] Daryl J Daley and Joseph Mark Gani. *Epidemic modelling: an introduction*. Number 15. Cambridge University Press, 1999.
- [19] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.
- [20] Khanh N Dinh and Roger B Sidje. Understanding the finite state projection and related methods for solving the chemical master equation. *Physical biology*, 13(3):035003, 2016.
- [21] Trang Dinh and Roger B Sidje. An adaptive solution to the chemical master equation using quantized tensor trains with sliding windows. *Physical Biology*, 17(6):065014, 2020.
- [22] Maciej Dobrzyński, Jordi Vidal Rodríguez, Jaap A Kaandorp, and Joke G Blom. Computational methods for diffusion-influenced biochemical reactions. *Bioinformatics*, 23(15):1969–1977, 2007.
- [23] Konstantin Dubrovinski and Martin Howard. Stochastic model for soj relocation dynamics in bacillus subtilis. *Proceedings of the National Academy of Sciences*, 102(28):9808–9813, 2005.
- [24] Johan Elf and Måns Ehrenberg. Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. *Systems biology*, 1(2):230–236, 2004.
- [25] Radek Erban, Jonathan Chapman, and Philip Maini. A practical guide to stochastic simulations of reaction-diffusion processes. *arXiv preprint arXiv:0704.1908*, 2007.
- [26] Silvio C Ferreira and Marcelo L Martins. Critical behavior of the contact process in a multiscale network. *Physical Review E*, 76(3):036112, 2007.

- [27] Zachary Fox, Gregor Neuert, and Brian Munsky. Finite state projection based bounds to compare chemical master equation models using single-cell data. *The Journal of chemical physics*, 145(7), 2016.
- [28] CW Gardiner, KJ McNeil, DF Walls, and IS Matheson. Correlations in stochastic theories of chemical reactions. *Journal of Statistical Physics*, 14:307–331, 1976.
- [29] Atiyo Ghosh, Andre Leier, and Tatiana T Marquez-Lago. The spatial chemical langevin equation and reaction diffusion master equations: moments and qualitative solutions. *Theoretical Biology and Medical Modelling*, 12:1–14, 2015.
- [30] Michael A. Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, 2000.
- [31] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics*, 22(4):403–434, 1976.
- [32] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.
- [33] Daniel T Gillespie. A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188(1-3):404–425, 1992.
- [34] Daniel T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58(1):35–55, 2007. PMID: 17037977.
- [35] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.
- [36] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.
- [37] Ankit Gupta, Jan Mikelson, and Mustafa Khammash. A finite state projection algorithm for the stationary solution of the chemical master equation. *The Journal of chemical physics*, 147(15), 2017.
- [38] Wolfgang Hackbusch. *Tensor spaces and numerical tensor calculus*, volume 42. Springer, 2012.
- [39] Johan Hattne, David Fange, and Johan Elf. Stochastic reaction-diffusion simulation with MesoRD. *Bioinformatics*, 21(12):2923–2924, 04 2005.
- [40] Markus Hegland and Jochen Garcke. On the numerical solution of the chemical master equation with sums of rank one tensors. *ANZIAM Journal*, 52:C628–C643, 2010.
- [41] Andreas Hellander, Michael J Lawson, Brian Drawert, and Linda Petzold. Local error estimates for adaptive simulation of the reaction–diffusion master equation via operator splitting. *Journal of computational physics*, 266:89–100, 2014.

- [42] Stefan Hellander, Andreas Hellander, and Linda Petzold. Reaction-diffusion master equation in the microscopic limit. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 85(4):042901, 2012.
- [43] Desmond J Higham. Modeling and simulating chemical reactions. *SIAM review*, 50(2):347–368, 2008.
- [44] Mark D Hill and Michael R Marty. Amdahl’s law in the multicore era. *Computer*, 41(7):33–38, 2008.
- [45] Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.
- [46] Samuel A Isaacson and David Isaacson. Reaction-diffusion master equation, diffusion-limited reactions, and singular potentials. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 80(6):066106, 2009.
- [47] Samuel A Isaacson and Ying Zhang. An unstructured mesh convergent reaction–diffusion master equation for reversible reactions. *Journal of Computational Physics*, 374:954–983, 2018.
- [48] Tobias Jahnke and Wilhelm Huisinga. Solving the chemical master equation for monomolecular reaction systems analytically. *Journal of mathematical biology*, 54:1–26, 2007.
- [49] Vladimir Kazeev, Mustafa Khammash, Michael Nip, and Christoph Schwab. Direct solution of the chemical master equation using quantized tensor trains. *PLoS computational biology*, 10(3):e1003359, 2014.
- [50] David G Kendall. On the generalized” birth-and-death” process. *The annals of mathematical statistics*, pages 1–15, 1948.
- [51] Boris N Khoromskij. Tensors-structured numerical methods in scientific computing: Survey on recent advances. *Chemometrics and Intelligent Laboratory Systems*, 110(1):1–19, 2012.
- [52] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.
- [53] Tamara Gibson Kolda. Multilinear operators for higher-order decompositions. Technical report, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA . . . , 2006.
- [54] Tatiana T Marquez-Lago and Kevin Burrage. Binomial tau-leap spatial stochastic simulation algorithm for applications in chemical kinetics. *The Journal of Chemical Physics*, 127(10):09B603, 2007.
- [55] Tatiana T. Marquez-Lago and Kevin Burrage. Binomial tau-leap spatial stochastic simulation algorithm for applications in chemical kinetics. *The Journal of Chemical Physics*, 127(10):104101, 2007.

- [56] Ralf Metzler. The future is noisy: the role of spatial fluctuations in genetic switching. *Physical Review Letters*, 87(6):068103, 2001.
- [57] Brian Munsky and Mustafa Khammash. The finite state projection algorithm for the solution of the chemical master equation. *The Journal of chemical physics*, 124(4):044104, 2006.
- [58] Ivan Oseledets and Eugene Tyrtyshnikov. Tt-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.
- [59] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.
- [60] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part I 23*, pages 521–536. Springer, 2012.
- [61] Nenad Pavin, Hana Čipčić Paljetak, and Vladimir Krstić. Min-protein oscillations in escherichia coli with spontaneous formation of two-stranded filaments in a three-dimensional stochastic reaction-diffusion model. *Physical Review E*, 73(2):021904, 2006.
- [62] Md Mustafijur Rahman and Roger B. Sidje. A study of a metapopulation model using the stochastic reaction diffusion master equation. In Henry Han and Erich Baker, editors, *Next Generation Data Science*, pages 242–253, Cham, 2024. Springer Nature Switzerland.
- [63] J Vidal Rodriguez, Jaap A Kaandorp, Maciej Dobrzyński, and Joke G Blom. Spatial stochastic modelling of the phosphoenolpyruvate-dependent phosphotransferase (pts) pathway in escherichia coli. *Bioinformatics*, 22(15):1895–1901, 2006.
- [64] Paul Sjoberg, Otto G Berg, and Johan Elf. Taking the reaction-diffusion master equation to the microscopic limit. *arXiv preprint arXiv:0905.4629*, 2009.
- [65] Stephen Smith and Ramon Grima. Breakdown of the reaction-diffusion master equation with nonelementary rates. *Physical Review E*, 93(5):052135, 2016.
- [66] MV Smoluchowski. Mathematical theory of the kinetics of the coagulation of colloidal solutions. *Z. Phys. Chem.*, 92:129–168, 1917.
- [67] Audrius B Stundzia and Charles J Lumsden. Stochastic simulation of coupled reaction–diffusion processes. *Journal of computational physics*, 127(1):196–207, 1996.
- [68] Jeroen S van Zon, Marco J Morelli, Sorin Tănase-Nicola, and Pieter Rein ten Wolde. Diffusion of transcription factors can drastically enhance the noise in gene expression. *Biophysical journal*, 91(12):4350–4367, 2006.

- [69] Jeroen S van Zon and Pieter Rein Ten Wolde. Simulating biochemical networks at the particle level and in time and space: Green's function reaction dynamics. *Physical review letters*, 94(12):128103, 2005.
- [70] M Alex O Vasilescu and Demetri Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I 7*, pages 447–460. Springer, 2002.
- [71] Melissa Vellela and Hong Qian. Stochastic dynamics and non-equilibrium thermodynamics of a bistable chemical system: the schlögl model revisited. *Journal of The Royal Society Interface*, 6(39):925–940, 2009.
- [72] Huy D Vo and Roger B Sidje. An adaptive solution to the chemical master equation using tensors. *The Journal of chemical physics*, 147(4), 2017.
- [73] Darren J Wilkinson. Stochastic modeling for quantitative description of heterogeneous biological systems. *Nature Reviews Genetics*, 10(2):122–133, 2009.
- [74] Stefanie Winkelmann and Christof Schütte. The spatiotemporal master equation: Approximation of reaction-diffusion dynamics via markov state modeling. *The Journal of Chemical Physics*, 145(21), 2016.
- [75] Verena Wolf, Rushil Goel, Maria Mateescu, and Thomas A Henzinger. Solving the chemical master equation using sliding windows. *BMC systems biology*, 4:1–19, 2010.
- [76] Zheming Zheng, Ryan M Stephens, Richard D Braatz, Richard C Alkire, and Linda R Petzold. A hybrid multiscale kinetic monte carlo method for simulation of copper electrodeposition. *Journal of Computational Physics*, 227(10):5184–5199, 2008.
- [77] Linhua Zhou and Meng Fan. Dynamics of an sir epidemic model with limited medical resources revisited. *Nonlinear Analysis: Real World Applications*, 13(1):312–324, 2012.