RICE UNIVERSITY

Multi-scale Behavior in Chemical Reaction Systems: Modeling, Applications, and Results

by

Jesse Hosea Turner III

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

APPROVED, THESIS COMMITTEE:

Dennis D. Cox, Director
Professor of Statistics

Steven J. Cox, Chairman
Professor of Computational and Applied Mathematics

Michael W. Fagan
Research Scientist, Department of Computational and Applied Mathematics

Houston, Texas

April, 2008
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI

UMI Microform 3309974
Copyright 2008 by ProQuest LLC.
All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 E. Eisenhower Parkway
PO Box 1346
Ann Arbor, MI 48106-1346
Abstract

Multi-scale Behavior in Chemical Reaction Systems: Modeling, Applications, and Results

by

Jesse Hosea Turner III

Four major approaches model the time dependent behavior of chemical reaction systems: ordinary differential equations (ODE's), the $\tau$-leap algorithm, stochastic differential equations (SDE's), and Gillespie's stochastic simulation algorithm (SSA). ODE's are simulated the most quickly of these, but are often inaccurate for systems with slow rates and molecular species present in small numbers. Under ideal conditions, the SSA is exact, but computationally inefficient. Unfortunately, many reaction systems exhibit characteristics not well captured individually by any of these methods. Therefore, hybrid models incorporating aspects from all four must be employed. The aim is to construct an approach that is close in accuracy to the SSA, useful for a wide range of reaction system examples, and computationally efficient.
The Adaptive Multi-scale Simulation Algorithm (AMSA) uses the SSA for slow reactions, SDE's for medium-speed reactions, ODE's for fast reactions, and the tau-leap algorithm for non-slow reactions involving species small in number. This article introduces AMSA and applies it to examples of reaction systems involving genetic regulation. A thorough review of existing reaction simulation algorithms is included. The computational performance and accuracy of AMSA’s molecular distributions are compared to those of the SSA, which is used as the golden standard of accuracy.

The use of supercomputers can generate much larger data sets than serial processors in roughly the same amount of computational time. Therefore, multi-processor machines are also employed to assess the accuracy of AMSA simulations.
Acknowledgements

The past six years have made for an eventful journey to say the least. The entire list of those who made it possible is too great for one page.

Special thanks goes out first to my parents whose encouragement and involvement can’t be described in words. Thanks to all grandparents here or in the next life for their positivity and high standards. Thanks, also to the committee members for their guidance, suggestions, and great attention to detail. Dennis Cox for his mathematical prowess and seemingly unlimited patience. Steve Cox for his dedication and inspiration for the project. Michael Fagan for his computational knowledge.

I thank Michael, Christy, and Brian (in order of age, not importance) for their liveliness and enthusiasm. Richard Tapia for his encouragement to go for the doctorate and not settle for the master’s, constructive criticisms of my talks, and efforts to help those in need. Theresa Chatman also deserves recognition for her efforts to reach into underrepresented populations for graduate school opportunities, roles in exciting conferences, and seemingly endless choice of good restaurants. Daria Lawrence for her support and aid with applications, regulations, and requirements.

I’d also like to express gratitude to the AGEP community: Mack, Kalatu, Big Ed, Little Ed, Kary, Jimmy, CJ, Clayton, etc. Thanks also goes to the extended Rice family: Mili, Fernando, Marco, Jay, Michelle, etc. The Houston family is included
as well: Garnet, Florence, Rashida, Nia, their families, Danny, Aunt Sylvia, their families, and God.
Contents

Abstract ii

Acknowledgements iv

List of Figures x

List of Tables xv

1 Introduction 1

1.1 Markov Processes ........................................ 3
  1.1.1 Mathematical Definition ............................ 3
  1.1.2 Illustrative Examples ............................. 4

1.2 Poisson Random Variables ............................ 12

1.3 The Diffusion Approximation .......................... 14

1.4 The Deterministic Model .............................. 16

2 A More Formal Review of Basic Methods 18

2.1 Notation ................................................. 19
2.1.1 An Illustrative Example ........................................... 21
2.1.2 Integral Representation of the Chemical State .............. 23

2.2 Deterministic Approaches ...................................... 25

2.3 Stochastic Approaches ......................................... 31

2.4 Exact Stochastic Simulation of Coupled Chemical Reactions .... 32
  2.4.1 Mathematical Derivation ...................................... 32
  2.4.2 The Algorithm ............................................... 37
  2.4.3 Gillespie Algorithm by Example ............................ 39

2.5 Improvements Upon Exact Stochastic Simulation ............... 40
  2.5.1 Efficient Exact Simulations .................................. 41

2.6 The τ-Leap Algorithm ........................................... 44

3 Review of Multi-scale Methods .................................. 47

3.1 Scale Coupling Approaches: Two-Scale Systems ............... 48
  3.1.1 The Haseltine and Rawlings Algorithm ....................... 48
  3.1.2 Dynamic Partitioning ........................................ 50
  3.1.3 Possibilities of More Than One Slow Reaction Per Time Step . 51

3.2 More Hybrid Methods: Using Leaping to Determine Time Steps ... 51
  3.2.1 The Maximal Time Step Method ............................. 52
  3.2.2 The Partitioned Leaping Algorithm .......................... 53

3.3 Consideration of Molecular Abundance ........................ 54
  3.3.1 The Method of Burrage, Tian, and Burrage .................. 54
3.3.2 The COAST Algorithm ........................................... 56

4 The Adaptive Multi-scale Simulation Algorithm ................. 59

4.1 The Adaptive Multi-scale Simulation Algorithm .......... 59

4.1.1 A Discussion of Thresholds .............................. 61

4.1.2 On Negative Populations ................................. 63

4.1.3 An Algorithmic Description ............................... 64

4.1.4 Uncoupling Slow Reactions ............................. 65

4.1.5 Important Considerations ................................. 68

4.2 AMSA vs. Similar Multi-scale Algorithms .................. 70

4.3 Applications .................................................. 71

4.3.1 A Simple Example: Genetic Self-Regulation .......... 72

4.3.2 A More Complicated Example: Detailed Genetic Regulation . 79

5 Large Scale Simulations Using Supercomputers ................. 83

5.1 Revisiting Genetic Regulation .................................. 84

5.1.1 The Self-Regulation Gene ............................... 85

5.1.2 Revisiting the lac System ............................... 89

5.2 Lambda Phage Regulation: A Viral Infection Model ........ 91

5.2.1 Modeling the Reaction System .......................... 92

5.2.2 Results .................................................. 94

6 Conclusion ....................................................... 102
6.1 On the Use of Multi-processor Machines .......................... 104
6.2 Future Directions ..................................................... 104

Bibliography ............................................................. 120
List of Figures

1.1 Time dependent behavior of A molecules in reaction $\dot{R} \rightarrow A$. The thick solid line is the discrete trajectory, while the thin dashed line is the limiting deterministic solution of the corresponding Poisson process. The following parameters were used: $c_1 = 8 \cdot 10^5 s^{-1}$, $X_R = 2 \cdot 10^5$. Throughout the plots in this chapter, a thin dashed line represents the limiting deterministic solution. ........................ 5

1.2 (Left Panel) Time dependent behavior for degradation reaction for discrete simulation (thick, solid) and corresponding exponential solution (thin, dashed). The following parameters were used: $c_2 = 1.5 s^{-1}$. (Right Panel) Trajectory for chemical system involving both creation and degradation of A ($c_1 = 8 \cdot 10^{-5} s^{-1}, c_2 = 1.5 s^{-1}, X_R = 2 \cdot 10^5$). 11

1.3 Plot of run similar to that of figure 1.1, but on a larger time scale. The trajectory begins to resemble a diffusion process rather than that of discrete jumps. .......................... 12
1.4 *(Left Panel)* Time dependent behavior for \( R \to C \) reaction using the diffusion approximation (solid line) and the discrete method (dash-dotted line). *(Right Panel)* Trajectories for degradation reaction. .... 13

1.5 Time dependent behavior for coupled reaction system using diffusion approximation (left panel) and discrete method (right panel). .... 16

2.1 Time dependent behavior of molecules in reaction (2.12) according to a deterministic approach. MATLAB’s *ode45* solver was used to solve the differential equations. .......... 28

2.2 Time dependent behavior of molecules in reactions from equations (2.16)-(2.18) according to deterministic model. .......... 30

2.3 A partition of the “jump” interval. Each subinterval has equal length \( \epsilon \). 33

2.4 *(Left Panel)* Time dependent behavior of molecules in \( A + B \to C \) reaction according to Gillespie’s algorithm (thick lines). The thin solid lines are results from the corresponding deterministic model. *(Right Panel)* Results from simulation of chemical reaction system described in equations (2.16)-(2.18). Again, Gillespie’s algorithm (thick lines) and the deterministic approach (thin lines) are compared. 40

4.1 Time courses for self-regulation model. The exact SSA is used here. .... 74

4.2 Time course for \( P \) according to SSA (top) and AMSA (bottom). .... 75

4.3 Bar plots illustrating the relative amount of time spent in each possible DNA state. .......... 76
4.4 Histograms of $R$ population for SSA (solid) vs. AMSA (dashed) for
$\Delta t = 0.0075$ (left) and $\Delta t = 0.1$. .......................... 77

4.5 Histograms of $P$ population for SSA (solid) vs. AMSA (dashed) for
$\Delta = 0.0075$ (left) and $\Delta t = 0.1$ (right). .................. 77

4.6 Confidence intervals for histograms of $R$ distribution for SSA (solid)
vs. AMSA (dashed) for $\Delta t = 0.0075$ (left) and $\Delta t = 0.1$ (right). .. 78

4.7 Confidence intervals for histograms of $P$ distribution for SSA (solid)
vs. AMSA (dashed) for $\Delta t = 0.0075$ (left) and $\Delta t = 0.1$ (right). .. 78

4.8 Time course for product molecules for $lac$ regulation system according
to one run each of the SSA (solid) and AMSA (dashed). This plot
represents ten generations. .............................................. 81

5.1 Bar plots illustrating the relative amount of time spent in each possible DNA state. .............................................. 87

5.2 Histogram of $R$ population for SSA (solid) vs. AMSA (dashed) with
$\tau = 0.0075$ (left panel) and $\tau = 0.1$ (right panel). ................. 87

5.3 Histogram of $P$ population for SSA (solid) vs. AMSA (dashed) with
$\tau = 0.0075$ (left panel) and $\tau = 0.1$ (right panel). ................. 88

5.4 Confidence intervals for histograms of $R$ distribution for SSA (solid)
vs. AMSA (dotted) with $\tau = 0.0075$ (left panel) and $\tau = 0.1$ (right panel). .............................................. 88
5.5 Confidence intervals for histograms of $P$ distribution for SSA (solid) vs. AMSA (dashed) with $\tau = 0.0075$ (left panel) and $\tau = 0.1$ (right panel). .................................................. 89

5.6 Time course for 100 realizations of the lac genetic regulation example (product molecules). SSA results are on the top, while those of AMSA are on the bottom. The graphs only represent one generation. 90

5.7 Molecular populations vs. time for instance in which CI dominates (left panel) and CRO dominates (right panel). The exact SSA is used for these simulations. ................................. 95

5.8 Molecular populations vs. time for instance in which CI dominates (left panel) and CRO dominates (right panel). AMSA is the used for these simulations. ................................. 95

5.9 Histogram of CI dimer populations after time $T=200s$ for the SSA (solid) and AMSA (dashed) with $\tau = 0.1$. ................................. 96

5.10 Histogram of CI dimer populations after time $T=200s$ for the SSA (solid) and AMSA (dashed) with $\tau = 10^{-4}$. ................................. 97

5.11 Histogram of CRO dimer populations after time $T=200s$ for the SSA (solid) and AMSA (dashed) with $\tau = 0.1$. ................................. 97

5.12 Histogram of CRO dimer populations after time $T=200s$ for the SSA (solid) and AMSA (dashed) with $\tau = 10^{-4}$. ................................. 98
5.13 (Left Panel) $L_1$ norm accuracy vs. time step for CI dimer. (Right Panel)

$L_1$ norm accuracy vs. time step for CRO dimer. .................................. 99

5.14 (Left Panel) Histogram of CI dimer population for modified system.

(Right Panel) Histogram for CRO dimer population. ............................... 101
List of Tables

2.1 Average CPU time comparison of original direct method and next reaction method on heat response shock model. Times are given in terms of seconds per simulation. Results are from Ref. [8]. ........ 44

3.1 Overview of multi-scale reaction system algorithms. The "***" indicates there's no abundance threshold. .. ......................... 48

3.2 Reaction propensity classification criteria for the partitioned leaping algorithm. [23] .............................................. 54

3.3 This table shows the algorithmic method used in the approach in Burrage et al. [4] for each combination of reaction speed (rows) and molecular abundance (columns). ......................... 55

4.1 This table shows the algorithmic method used in AMSA for each combination of reaction speed (rows) and molecular abundance (columns). ........................................ 61

4.2 Parameters for toy model simulation. ......................... 74
4.3 Initial values for toy model simulation. .......................... 75
4.4 Histogram distance between AMSA and SSA for R and P distributions. 79
4.5 Parameters for reaction system examples. .......................... 79
4.6 Analysis of run times for SSA and Adaptive Multi-scale models. ... 79
4.7 Analysis of run times for SSA and Adaptive Multi-scale models for Kierzek's lactose regulation system. .......................... 81
4.8 Reactions and parameters for lac system. The symbol $\theta$ denotes a degradation reaction. .......................... 82
4.9 Initial values for Kierzek model. .......................... 82
5.1 Histogram distance between AMSA and SSA for R and P distributions. 87
5.2 Parameters for lambda phage model. For DNA states, $D$ indicates all three operator regions are unbound. $Di$ and $Do$ represent that operator regions are occupied by $CI$ and $CRO$, respectively. The subscript (e.g. the 2 in $Di_2$) indicates the total number of $CI$ or $CRO$ molecules are attached to the operator regions. The symbol $\theta$ denotes a degradation reaction. .......................... 93
5.3 Initial values for lambda phage model. .......................... 94
5.4 Parameters for lambda phage system. .......................... 94
5.5 Analysis of run times for 1000 runs of viral infection system. Simulations were run serially. .......................... 99
5.6 Parameters for modified lambda phage reaction system. ............ 100
5.7 Initial values for modified lambda phage model. . . . . . . . . . . . . 101

5.8 Analysis of run times for 1000 runs of modified viral infection system.

Simulations were run serially. . . . . . . . . . . . . . . . . . . . . . . . . 101
Chapter 1

Introduction

Calculating the molecular counts of a spatially homogeneous, well-mixed chemical reaction system has been the subject of decades of research. Though there is an abundance of methods, there isn’t a model that’s optimal for all situations. Many factors must be taken into account to determine what is best for the reaction system under study, especially accuracy and computational efficiency.

Deterministic approaches to this problem are not only the most widely used methods, but also have been around the longest. Strongly based on the chemical law of mass action, these approaches use ordinary differential equations to solve for the time evolution of a reaction system. The stochastic approach depends on the stochastic formulation of chemical kinetics. This approach is based on reaction rates that determine the probabilities of reactions occurring in infinitesimally small intervals. From this, a “chemical master equation” is derived, which gives the
probability distribution of the molecular counts at each time. However, it is typically not practical to solve this equation. Thus, simulation-based methods have become the usual avenues for solving these problems.

Each approach has its advantages and disadvantages. Often enough, deterministic methods are sufficient, due to their computational speed, and the numerous analytical tools available to examine ODE's. However, at times, the underlying physical assumptions of a deterministic approach are not met, accuracy issues arise, and the stochastic formulation is the ideal choice. Sometimes, neither is suitable, and a hybrid treatment incorporating elements of both approaches is needed.

It should be emphasized that both avenues are not exactly distinct. In fact, if one keeps the chemical concentrations constant, but increases the number of molecules and container volume to infinity, the stochastic formulation approaches the law of mass action. [32] For this reason, ODE model's are often sufficient for systems where all reactions occur frequently and all molecular types are abundant.

Here, the existing methods used to simulate time trajectories of a reaction system are examined. Some rely strictly on one type of model, while others are multi-scale and employ a mixture of models. A new hybrid model is then introduced that emphasizes both speed and accuracy. Gillespie's stochastic simulation algorithm is taken to be the standard against which to measure accuracy. [17] Also included is a discussion on the underlying mathematical theory and how each model is related analytically.
The underlying theory for multi-scale simulation relies on four basic tools: discrete Markov processes, Poisson random variables, stochastic differential equations, and ODE's. This chapter provides a brief review on the tools using simple examples. Discussion on the relations among the tools is provided as well. The following chapter gives a more rigorous analysis.

1.1 Markov Processes

A Markov process is one in which the probability of a future event conditioned on the past and present is only dependent on the present. The present state of a Markov process is the current value of the process. Real life examples that can be accurately modeled by Markov processes include the amount of offspring from a particular ancestor and the number of people waiting in line at a store.

1.1.1 Mathematical Definition

Now, Markov processes will be defined mathematically. A real-valued, random process \( X(t) \) is Markov if, for a finite set of times \( t_0 < t_1 < \ldots < t_n < t_{n+1} \) and any set \( B \),

\[
P[X(t_{n+1}) \in B | X(t_0) = x_0, X(t_1) = x_1, \ldots, X(t_n) = x_n] = P[X(t_{n+1}) \in B | X(t_n) = x_n].
\]
Notice how the values of $X$ from $t_0$ up to $t_{n-1}$ have no bearing on $X(t_{n+1})$. For this reason, Markov processes are called memoryless. This property makes the underlying mathematics more manageable.

If the times are discrete, the process is also called a Markov chain. The transition probabilities associated with Markov chains are easily represented by a matrix, even further simplifying the mathematics. Examples of Markov chains include the amount of money a gambler has after successive rolls of die and the random walk.

It is assumed the present state of a chemical system, defined as the vector of quantities of chemical species involved, can be described by a Markov process, since the next state of the system only depends on the present amount of each species.

### 1.1.2 Illustrative Examples

Consider the chemical reaction

$$\bar{R} \xrightarrow{c_1} A.$$ 

The bar over the $R$ indicates that there is an unlimited supply available. Therefore, one can assume the amount of $R$ to be constant. This situation is similar to having a large reservoir of molecules that emits $A$ at a constant rate $c_1$. The number of $A$ molecules with respect to time, denoted by $X_A(t)$ is a Markov process: given $X_A(t)$, the probability density of $X_A$ at a future time $t + s$ only depends on the present amount of molecules $X_A(t)$. A sample trajectory for $X_A(t)$ is presented in figure 1.1.
Figure 1.1: Time dependent behavior of A molecules in reaction $\hat{R} \to A$. The thick solid line is the discrete trajectory, while the thin dashed line is the limiting deterministic solution of the corresponding Poisson process. The following parameters were used: $c_1 = 8 \cdot 10^5 s^{-1}$, $X_R = 2 \cdot 10^5$. Throughout the plots in this chapter, a thin dashed line represents the limiting deterministic solution.
In order to proceed, the transition probabilities for the reaction must be obtained. For this reaction, the probability that a reaction proceeding at a rate $c_1$ occurs in the time interval $[t, t + \Delta t)$ when there are $X_R$ reactants present at time $t$ is given by

$$P[X_A(t + \Delta t) = X_A(t) + 1|X_A(t)] = c_1 X_R \Delta t + o(\Delta t). \quad (1.1)$$

There is always a possibility the reaction happens more than once in a time interval; however, the probability this occurs shrinks with the length of the time interval $\Delta t$ on a smaller order than $\Delta t$. Hence, the $o(\Delta t)$ term is placed in (1.1). Dividing both sides by $\Delta t$ and taking the limit as the interval becomes infinitesimally small yields the rate $a_1(X_A, X_R) = c_1 X_R = a_1$. These rates are also called reaction propensities.

With the transition rate in hand, the Chapman-Kolmogorov equation [46] can be derived. Consider the probability density

$$P[X_A(t) = x] \equiv p(x, t). \quad (1.2)$$

There are two ways to arrive at the state $X_A(t) = x$. One possibility is that there are $x - 1$ molecules and the reaction occurs once. The probability of this event is $a_1 \Delta t + o(\Delta t)$. The other possibility is that there are already $x$ molecules of $C$ at time $t$ and no reaction occurs. The probability that this occurs is $1 - a_1 \Delta t + o(\Delta t)$. 
Therefore, the law of total probability gives an expression for (1.2):

\[ p(x, t + \Delta t) = a_1 p(x - 1, t)\Delta t + (1 - a_1\Delta t)p(x, t) + o(\Delta t). \]  

(1.3)

This is the Chapman-Kolmogorov equation for the reaction \( R \to A. \)

The transformation of equation (1.3) is simple. Subtracting \( p(x, t) \) from both sides yields

\[ p(x, t + \Delta t) - p(x, t) = a_1 p(x - 1, t)\Delta t - a_1 \Delta t p(x, t) + o(\Delta t). \]

Dividing both sides by \( \Delta t \) and taking the limit as \( \Delta t \to 0 \) gives the following:

\[ \frac{\partial p(x, t)}{\partial t} = a_1 p(x - 1, t) - a_1 p(x, t). \]  

(1.4)

Equation (1.4) is termed the chemical master equation (CME). [46, 40, 38] The CME is just the differential equations form of the Chapman-Kolmogorov equation. [46] Caution must be exercised when interpreting the CME. Given an initial time \( t_0 \) and an arbitrary state \( x_0 \), the solution of equation (1.4) is

\[ p(x, t) = \frac{e^{-a_1(t-t_0)}[a_1(t-t_0)]^{(x-x_0)}}{(x-x_0)!}, \]  

(1.5)

the Poisson pdf. Substitution of (1.5) into equation (1.4) should verify this.

The ODE in (1.4) describes the probability density function for what is known
as the Poisson process. It has three important properties: (a) the number of jumps \( q \) for non-overlapping intervals are independent, (b) the probability \( q = 1 \) in a sufficiently small interval \((t, t + \Delta t)\) is \( a_1 \Delta t + o(\Delta t)\), and (c) the probability \( q > 1 \) in \((t, t + \Delta t)\) is of \( o(\Delta t) \). The simplicity of this example makes its corresponding master equation solvable. Explicit solutions to the CME for larger reaction systems are much more difficult to find.

It’s often useful to get a sense of the “average” behavior for a chemical trajectory. Through manipulation of the master equation, one can find the limiting solution of \( X_A(t) \). Multiplying of both sides of (1.4) by \( x \) and summing from 0 to \( \infty \),

\[
\frac{\partial}{\partial t} \sum_{x=0}^{\infty} x p(x, t) = \sum_{x=0}^{\infty} a_1 x p(x - 1, t) - \sum_{x=1}^{\infty} a_1 x p(x, t).
\]

After re-indexing the first term on the right hand side,

\[
\frac{\partial}{\partial t} \sum_{x=0}^{\infty} x p(x, t) = \sum_{x=0}^{\infty} a_1 (x + 1) p(x, t) - \sum_{x=0}^{\infty} a_1 x p(x, t).
\]

Cancellation on the right hand side yields

\[
\frac{\partial \mu_A(t)}{\partial t} = a_1,
\]

where \( \mu_A(t) = \sum_{x=0}^{\infty} x p(x, t) \). The solution to the ODE in (1.6), found through simple integration, is \( \mu_A(t) = \mu_A(0) + a_1 t \). The figures in this chapter all include the limiting
deterministic solution given by the law of mass action for illustration purposes. It's related to the chemical law of mass action. In many examples, this is in fact the mean of the stochastic model. This is used often for chemical systems with large numbers of molecules and fast reaction rates. More on deterministic approaches to chemical reactions will be discussed in later chapters.

The decay process also lends itself to Markov process modeling. The chemical reaction

$$A \xrightarrow{c_2} \emptyset$$

(1.7)

describes the loss of $A$ molecules without replenishment. Figure 1.2 gives a sample trajectory for a given choice of $c_2$. The transition probability is as follows:

$$P[X_A(t + \Delta t) = X_A(t) - 1|X_A(t)] = c_2 X_A(t) \Delta t + o(\Delta t).$$

This time, the transition probability does depend on time. The reaction propensity is given by $a_2(X_A(t), X_B(t)) = c_2 X_A(t)$. In the previous example, there was a constant amount of reactant. In the decay process, the amount of reactant has an effect on the decay rate, since fewer molecules mean a slower reaction speed. The time dependency of the rate keeps the process from being Poissonian. Better yet, the mean of several runs appears to be exponential.
The master equation for the reaction for (1.7) is

\[ \frac{\partial p(x,t)}{\partial t} = c_2(x + 1)p(x + 1, t) - c_2xp(x, t). \]

Note the similarities to that of the first example.

The derivation of the ODE for the mean function follows a similar procedure to the first example:

\[
\begin{align*}
\frac{\partial}{\partial t} \sum_{x=0}^{\infty} xp(x,t) &= \sum_{x=0}^{\infty} c_2(x^2 + x)p(x + 1, t) - \sum_{x=0}^{\infty} c_2x^2p(x, t). \\
\frac{\partial}{\partial t} \sum_{x=0}^{\infty} x^2p(x,t) &= \sum_{x=1}^{\infty} c_2(x^2 - x)p(x, t) - \sum_{x=0}^{\infty} c_2x^2p(x, t). \\
\frac{\partial \mu_A(t)}{\partial t} &= -c_2\mu_A(t).
\end{align*}
\]

The solution to this well-known ODE is \( \mu_A(t) = \mu_A(0)e^{-c_2t} \), the general form of exponential decay.

Now, consider the system combining both types of reactions:

\[ R \xrightarrow{c_1} A \] (1.8)

\[ A \xrightarrow{c_2} \emptyset \] (1.9)

In this case, \( X_A(t) \) will settle into a steady-state behavior with fluctuations. Figure 1.2 provides a time graph of \( X_A(t) \) for the coupled reactions. Parameters and
reaction constants are kept the same as in the previous examples.

Reaction firings are more frequent. Assuming rates remain the same, more reactions imply smaller time steps between events. The master equation is given by

$$\frac{dp(x,t)}{dt} = c_1 X_R p(x-1,t) + c_2 (x+1)p(x+1,t) - (c_1 X_R + c_2 x)p(x,t).$$

The limiting solution for the coupled reactions can be found using procedures similar to those of the first two examples. Since the derivations are similar, only the ODE for the mean function is listed here:

$$\frac{d\mu_A(t)}{dt} = c_1 X_R - c_2 \mu_A(t).$$

Simulation of small chemical systems reaction by reaction is satisfactory; but for
Figure 1.3: Plot of run similar to that of figure 1.1, but on a larger time scale. The trajectory begins to resemble a diffusion process rather than that of discrete jumps.

larger systems, it's computationally inefficient. The following section introduces an effective approximation that takes less time to compute.

1.2 Poisson Random Variables

Reconsider the reaction

\[ \mathcal{R} \xrightarrow{\omega} A. \]

What would happen if \( X_A(t) \) was examined in the context of a larger time scale? In figure 1.1, the scaling was such that each individual jump appeared to be important. If \( X_A(t) \) was examined on a larger time scale, though, the behavior of the same process would have a different appearance. Figure 1.3 contains graphs of \( X_A(t) \)
Figure 1.4: (Left Panel) Time dependent behavior for $R \rightarrow C$ reaction using the diffusion approximation (solid line) and the discrete method (dash-dotted line). (Right Panel) Trajectories for degradation reaction.

on a larger time scale. Instead of Poisson jumps, the function resembles a linear function with noise.

The present objective is to find this "noisy" approximation to the discrete jump process. The state of the system at a future time $t + \Delta t$ can be represented by the following equation:

$$X_A(t + \Delta t) = X_A(t) + K(X_A(t), \Delta t),$$  \hspace{1cm} (1.10)

where $K(X_A(t), \Delta t)$ is some random variable. The Poisson distribution gives the number of events that occur for an arbitrary time interval, assuming the time in between events is exponentially distributed. More specifically, the Poisson random variable $P(a_1, \Delta t)$ yields how many times the reaction $R \rightarrow A$ occurs in the interval
Using this information, equation (1.10) can be rewritten

\[ X_A(t + \Delta t) = X_A(t) + \mathcal{P}(a_1, \Delta t). \]

This substitution is exact for this example. In general, it works provided the time step \( \Delta t \) is small enough to keep the probability \( a_1 \Delta t \) from changing significantly, but large enough that the reaction occurs several times.

1.3 The Diffusion Approximation

By way of Stirling’s approximation, it can be proven that

\[ \mathcal{P}(a_1, \Delta t) \approx \mathcal{N}(a_1 \Delta t, a_1 \Delta t), \]

where \( \mathcal{N}(\mu, \sigma^2) \) is a normal random variable with mean \( \mu \) and variance \( \sigma^2 \). [18] So (1.10) can again be rewritten

\[ X_A(t + \Delta t) \approx X_A(t) + \mathcal{N}(a_1 \Delta t, a_1 \Delta t). \]

The linear combination theorem implies \( \mathcal{N}(\mu, \sigma^2) = \mu + \sigma \mathcal{N}(0, 1) \). After making one final substitution,

\[ X_A(t + \Delta t) - X_A(t) = a_1 \Delta t + \sqrt{a_1 \Delta t} \mathcal{N}(0, 1). \]
This is the discrete version of a stochastic differential equation (SDE). [46] In chemical literature, it's called the chemical Langevin equation. Gillespie derives the general approximation for an arbitrary chemical system in an article discussing this equation. [18] The derivations of the SDE for the reactions in (1.7) and (1.8) are analogous.

Stochastic differential equations have long been used to model random processes. [39] An SDE is defined to be a differential equation of the form

\[
\frac{dX(t)}{dt} = b(X(t), t) + \phi(X(t), t) \frac{dW(t)}{dt}.
\] (1.11)

\(W(t)\) is a white noise process, while \(b\) and \(\phi\) are \(\mathbb{R}\)-valued functions. The drift term \(b\) gives the limiting behavior of the system; the diffusion term \(\phi\) is responsible for the fluctuations. When \(\phi = 0\), the SDE becomes strictly an ordinary differential equation (ODE). The quantity \(\frac{dW(t)}{dt}\) actually doesn't exist and is presented in this form as a matter of convenience. The more accurate form of (1.11) is

\[
dX(t) = b(X(t), t)dt + \phi(X(t), t)dW(t).
\] (1.12)

The solution \(X(t)\) to (1.12) is called a diffusion process. Figure 1.5 compares the diffusion approximation for the system in (1.8) to that of the discrete method.

Figure 1.4 gives the SDE approximations for the first two reaction examples on an even larger scale. Notice that in both cases, the discrete trajectories don’t
Figure 1.5: Time dependent behavior for coupled reaction system using diffusion approximation (left panel) and discrete method (right panel).

coincide with those of the SDE approach. This illustrates a dilemma in examining stochastic simulations. Two runs of a stochastic process will almost never coincide due to the randomness involved. Better ways of analyzing stochastic runs include comparing the averages of many runs and computing summary statistics.

Diffusion processes are important in molecular simulations, because of their mediating role. They have the continuous nature of ODE's while obtaining the stochasticity of probabilistic discrete processes. Also, it usually takes less computational time to simulate a solution to an SDE than a discrete process. Future chapters allot more attention to diffusion approximations.

1.4 The Deterministic Model

In equation 1.12, large reactant populations and fast rates can make $b(X(t), t)dt$ much larger than $\phi(X(t), t)$. The second term would then become insignificant
when compared to the first. Since generation of the normal random number can be computationally expensive, deletion of the second term is more beneficial:

\[ dX(t) = b(X(t), t)dt. \]  \hspace{1cm} (1.13)

This differential equation represents the well-known chemical law of mass action. The stochastic fluctuations seen in the previous approaches aren't needed here. An important topic in stochastic simulation models is determining when to use which approach.

This chapter succinctly summarizes the basic methods with which to model reaction systems. However, this was done informally with simple examples. In order to gain a better understanding of multi-scale methods, one needs more generality and a detailed analytical framework for these basic approaches. A more formal introduction awaits in the following chapter.
Chapter 2

A More Formal Review of Basic Methods

The past couple of decades have brought an abundance of new approaches to chemical reaction simulations. Some are more accurate while others are more computationally efficient. Deterministic models based on the chemical law of mass action were initially used to project time trajectories of chemical species. Gillespie [17] spearheaded the most recent flurry of stochastic simulation models with his exact stochastic simulation algorithm (SSA). Later, Gillespie introduced the $\tau$-leap algorithm in an effort to speed up reaction simulations. This chapter provides brief explanations and derivations of the foundational models. First, a primer on notation is given.
2.1 Notation

This section describes specific notations used in subsequent sections. Consider molecular species $A_1, A_2, ..., A_N$ undergoing $M$ chemical reactions indexed by $j$:

$$a_{j1}A_1 + a_{j2}A_2 + ... + a_{jN}A_N \xrightarrow{c_j} \kappa_{j1}A_1 + \kappa_{j2}A_2 + ... + \kappa_{jN}A_N,$$  \hspace{1cm} (2.1)

where $a_{ji}$ is the number of $A_i$ molecules consumed in reaction $j$ and $\kappa_{ji}$ is the number of molecules created. For a fixed time interval $[t, t + \tau)$, the probability this reaction occurs for a given number of reactant molecules is

$$a_j \tau + o(\tau),$$  \hspace{1cm} (2.2)

where $a_j$ is the reaction propensity. The $a_j$'s are defined to be

$$a_j \equiv c_j \begin{pmatrix} X_1 \\ \alpha_{j1} \\ \vdots \\ \alpha_{jN} \end{pmatrix} = \begin{pmatrix} X_2 \\ \alpha_{j2} \\ \vdots \\ \alpha_{jN} \end{pmatrix},$$  \hspace{1cm} (2.3)

where

$$\begin{pmatrix} X_i \\ \alpha_{ji} \end{pmatrix} \equiv \frac{X_i(X_i - 1)...(X_i - \alpha_{ji} + 1)}{\alpha_{ji}!}.$$  \hspace{1cm} (2.4)

$X_i$ is just the amount of $A_i$ molecules at a given time. The expression $h_j$ on the right hand side of (2.3) is a reflection of the possible number of combinations of reactant
molecules. For instance, if there is one of each reactant (e.g. $A + B \rightarrow C$), then

$$h_j = \begin{pmatrix} X_A \\ 1 \end{pmatrix} \begin{pmatrix} X_B \\ 1 \end{pmatrix} = X_A X_B.$$  

As intuition suggests, if there are $X_A$ molecules of $A$ and $X_B$ molecules of $B$, there are $X_A X_B$ possible combinations for a collision of the two. If the two of the same reactant type are colliding (e.g. $2A \rightarrow B$, then

$$h_j = \begin{pmatrix} X_A \\ 2 \end{pmatrix} = \frac{X_A (X_A - 1)}{2}.$$  

This is a bit more subtle. There are $X_A$ choices for the first molecule; as a result, there are $X_A - 1$ choices for the second molecule. The product should be divided by two to prevent recounting. The extension to situations in which there are three or more reactants is trivial, though such occurrences are rare.

From henceforth, the state of the system is denoted with the row vector $X(t) = (X_1(t), X_2(t), \ldots, X_N(t))$. It’s assumed the molecules are well-mixed and uniformly distributed in a container of volume $V$. The stochastic rate constant $c_j$ indicates a sense of how fast reaction $j$ fires. The reaction rate constant depends on many
physical factors including volume, mean velocities of the molecules, molecular
diameter, and temperature. Temperature and mean velocity are both averages
taken over the entire volume of the container. As a result, one can easily see why
spatial homogeneity and a well-mixed system are important assumptions. Because
c_j differs a bit from the deterministic rate often given in chemical literature, further
explanation is given below. The combinatorial quantity in (2.4) is the total number
of different combinations of reactant molecules that could produce the jth reaction.

At times, it’s more convenient to keep track of a reaction’s overall effect on the
system state. This is accomplished by introducing a stoichiometric matrix ν that
subtracts the number of reactants from the number of products. More specifically,
for every i and j,

\[ ν_{ji} = κ_{ji} - α_{ji}. \]

The jth row of ν will simply be denoted as ν_j.

It should be noted that the theory in this section follows the stochastic formulation
of chemical kinetics. The deterministic formulation is mentioned later.

2.1.1 An Illustrative Example

Consider the reaction system
\[ A_1 + A_2 \rightarrow A_3 \]
\[ 2A_2 \rightarrow A_1 \]  \hspace{1cm} (2.5)
\[ A_3 \rightarrow 3A_1. \]

The corresponding stoichiometric reactant matrix \( \alpha \) is
\[
\alpha = \begin{pmatrix} 
1 & 1 & 0 \\
1 & 0 & 0 \\
0 & 1 & 1 \\
0 & 0 & 1 
\end{pmatrix}.
\]

Analogously, the stoichiometric product matrix is
\[
\kappa = \begin{pmatrix} 
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
3 & 0 & 0 
\end{pmatrix}.
\]

The matrix \( \nu \) for the chemical reaction system in (2.5) is
\[
\nu = \begin{pmatrix} 
-1 & -1 & 1 \\
1 & -2 & 0 \\
3 & 0 & -1 
\end{pmatrix}.
\]
If the state of the system at \( t_i \) is \( X(t_i) \) and the next reaction \( j \) occurs at time \( t_{i+1} \), then

\[
X(t_{i+1}) = X(t_i) + \nu_j.
\]

It should be stressed that all three matrices are important in the quantitative analysis of reaction systems. Keeping track only of \( \nu \) can hide certain aspects of the example under study (e.g. catalytic reactions). The entries of \( \alpha \) and \( \kappa \) are limited to nonnegative integers, while those of \( \nu \) can be any integer.

### 2.1.2 Integral Representation of the Chemical State

When considering time trajectories of a reaction system, it's occasionally more benign to adopt an integral representation. Given a start time of \( t_0 = 0 \), the number of times reaction \( j \) has occurred by time \( t \) is

\[
Y_j\left(\int_0^t a_j(X(s))ds\right), \quad j = 1, \ldots, M,
\]

where the \( Y_j \)'s are independent unit Poisson processes. Reaction channels are the only manner by which the state vector \( X(t) \) changes. Thus, using the update matrix \( \nu \), a simple equation relates \( X(t) \) to these processes:

\[
X(t) = X(0) + \sum_{j=1}^{M} \nu_j Y_j\left(\int_0^t a_j(X(s))ds\right).
\]
The time evolution of system (2.5) in integral form is given by

\[ X_1(t) = X_1(0) - Y_1 \left( c_1 \int_0^t X_1(s)X_2(s)ds \right) + Y_2 \left( c_2 \int_0^t \frac{1}{2}(X_2(s))(X_2(s) - 1)ds \right) + 3Y_3 \left( c_3 \int_0^t X_3(s)ds \right) \]

\[ X_2(t) = X_2(0) - Y_1 \left( c_1 \int_0^t X_1(s)X_2(s)ds \right) - 2Y_2 \left( c_2 \int_0^t \frac{1}{2}(X_2(s))(X_2(s) - 1)ds \right) \]

\[ X_3(t) = X_3(0) + Y_1 \left( c_1 \int_0^t X_1(s)X_2(s)ds \right) - Y_3 \left( c_3 \int_0^t X_3(s)ds \right). \]

Similar expressions can be derived for strictly deterministic models.

When the system at hand is modeled using multi-scale methods, the assumption that all \( Y_j \)'s are unit Poisson processes should be relaxed. For example, a situation could arise in which a particular reaction \( j \) is better suited for a law of mass action treatment. Then, the following approximation results:

\[ Y_j \left( \int_0^t a_j(X(s))ds \right) \approx \int_0^t a_j(X(s))ds. \]

Chapter 3 will examine the situations in which it's safe for \( Y_j \) to be continuous. Chapter 4 uses the notation in (2.6) to connect two different hybrid algorithms. The following sections give brief summaries to the existing chemical reaction system models as well as some illustrative examples.
2.2 Deterministic Approaches

The traditional way of simulating molecular populations in a reaction system involves coupled ordinary differential equations. [17] Assume there are N different types of molecules undergoing M different possible reactions. If the concentration of chemical specie \( S_i \) at time \( t \) is denoted \( X_i(t) \), then the coupled ODE system

\[
dX_1(t)/dt = \sum_{j=1}^{M} v_{ij} k_j \prod_{q=1}^{N} [X_q]^{a_{ij}}
\]

\[
dX_2(t)/dt = \sum_{j=1}^{M} v_{ij} k_j \prod_{q=1}^{N} [X_q]^{a_{ij}}
\]

\[
dX_3(t)/dt = \sum_{j=1}^{M} v_{ij} k_j \prod_{q=1}^{N} [X_q]^{a_{ij}}
\]

\[
dX_N(t)/dt = \sum_{j=1}^{M} v_{ij} k_j \prod_{q=1}^{N} [X_q]^{a_{ij}}
\]

is used to model the time dependent behavior of \( X_i(t) \). The equations in (2.7) are typically nonlinear. A particular set of initial conditions gives a unique time trajectory. Thus, as the description "deterministic" suggests, at a particular time, the amount of each chemical concentration is predetermined by the initial concentrations - there's no uncertainty.

These equations follow the deterministic formulation of chemical kinetics. The expressions on the right hand side of (2.7) are very similar to the entity \( a_j \) mentioned
in section 2.1. Note the [ ] symbol that is used for molar concentrations. \([X_i](t)\) and \(X(t)\) can be related by

\[
[X_i](t) = \frac{X(t)}{N_A V},
\]  

(2.11)

where \(N_A\) is Avogadro's number. The deterministic formulation of chemical kinetics uses concentrations, while the stochastic formulation counts individual molecules.

Also note the use of the rate constant \(k_j\). This "bulk" rate constant is closely related to the one used in the stochastic formulation \(c_j\). The two can be related with the expression:

\[
c_j = k_j \left( \prod_{q=1}^{N} \alpha_{q!} \right) V^{1-\sum_{i} v_i}.
\]

If the reaction is unimolecular (e.g. \(A_1 \rightarrow A_2\)), this relation simplifies to

\[
k_j = c_j.
\]

For reactions involving two distinct reactants (e.g. \(A_1 + A_2 \rightarrow A_3\)),

\[
k_j = Vc_j.
\]

When the reaction involves two identical reactants (e.g. \(A_1 + A_1 \rightarrow C\)),

\[
k_j = Vc_j/2.
\]
Consider the following simple reaction:

\[ A + B \overset{k}{\rightarrow} C. \]  

(2.12)

The corresponding set of ODE's is as follows:

\[
\frac{d[A](t)}{dt} = -k[A](t)[B](t) \tag{2.13}
\]

\[
\frac{d[B](t)}{dt} = -k[A](t)[B](t) \tag{2.14}
\]

\[
\frac{d[C](t)}{dt} = k[A](t)[B](t) \tag{2.15}
\]

Equations (2.13) and (2.14) have minus signs, since the reaction results in the disappearance of A and B.

Figure 2.1 shows the results of a simulation of the above system. ODE's are typically solved numerically using scientific programming packages such as FORTRAN and MATLAB. MATLAB's \texttt{ode45} command was employed to solve the corresponding equations. Because numerical solvers are relatively fast, deterministic models are often more computationally efficient than others.

At times, the ODE's for chemical reaction systems aren't as simple as in the above example. Consider the more complicated example
Figure 2.1: Time dependent behavior of molecules in reaction (2.12) according to a deterministic approach. MATLAB’s ode45 solver was used to solve the differential equations.

\[ A \xrightarrow{k_1} B, \]  
\[ B + E \xrightleftharpoons[k_2]{k_3} BE \xrightarrow{k_3} C + E, \]  
\[ A + C \xrightarrow{k_4} 2A. \]

Let it be noted the middle reaction is assumed to be enzymatic. The Michaelis-Menten [27] approximation yields an equation describing the change in the amount
of substrate B:

$$\frac{d[B_{sub}]}{dt} = -k_3 \frac{[B][E]}{K_2 + [B]}$$

where $K_2 = \frac{k_2 + k_3}{k_2}$. The following ODE set describes the system as a whole:

$$\frac{d[A]}{dt} = -k_1[A] + k_4[A][C]$$

$$\frac{d[B]}{dt} = k_1[A] - k_3 \frac{[B][E]}{K_2 + [B]}$$

$$\frac{d[C]}{dt} = k_3 \frac{[B][E]}{K_2 + [B]} - k_4[A][C].$$

The quantity $[E]$ is assumed to be relatively constant throughout the simulation. Thus, it's not necessary to derive a corresponding ODE. Figure 2.2 shows the results from applying the deterministic approach to this example.

Another advantage of deterministic approaches is the wide array of mathematics than can be applied to them. Tools such as numerical optimization can be used to extract parameters from the ODE's. [10] This has great practical use in biology as the in vivo rates for many cellular reactions have yet to be uncovered. Reverse engineering may also be employed to obtain approximate values for biological entities such as Hill coefficients. [35]

Time delays can be implemented for phenomena that occurs after a lapse in time. Mahaffy uses delayed differential equations to study oscillations in lac enzyme
Figure 2.2: Time dependent behavior of molecules in reactions from equations (2.16)-(2.18) according to deterministic model.

production. [36] Dynamical systems studies have been utilized to help determine the conditions under which these oscillations exist. [48]

Though deterministic modeling has proven an effective vehicle for projecting chemical time evolution, it leaves much to be desired. The physical dynamics of a chemical reaction are seriously altered when there are a small number of molecules involved. In addition, the continuous curves resulting from ODE solutions don’t resemble noisy natural data. How does one account for this? The next section offers alternate ways for modeling chemical systems.
2.3 Stochastic Approaches

The dynamics of a biochemical reaction system can change significantly depending on the number of molecules involved. For instance, in some settings, certain molecules may be small in number, but have a large effect on the system. A prime example is genetic regulation. Whether the gene is on or off can have a significant bearing on the cell's properties. Deterministic approaches aren't ideal for such situations, since an underlying assumption is that the system have sufficiently large molecular populations.

This is where stochastic simulation modeling can be effective. Gillespie [16] introduced a Monte Carlo algorithm that projected a chemical reaction system's time evolution by simulating which reactions occurred at what times. There were previous ventures into stochastic approaches for chemical reaction modeling (see [37], [11], [26]). However, they were based directly on the chemical master equation (CME). The CME is difficult to solve directly; therefore, the practical insight given by those publications is limited. Gillespie's algorithm is rigorously equivalent to the CME [8]. Furthermore, it was derived without solving the CME directly.
2.4 Exact Stochastic Simulation of Coupled Chemical Reactions

Gillespie's algorithm, also known as the stochastic simulation algorithm, simulates the time trajectories of the molecular populations in a chemical reaction system by choosing

- which reaction occurs next, and
- when this reaction occurs.

After implementing the above, the amount of each molecular type are reset accordingly, and the algorithm repeats itself.

2.4.1 Mathematical Derivation

Gillespie's stochastic simulation algorithm not only is equivalent to the CME, but also gives a better idea of how a chemical reaction system behaves in one run. One can get a sense of how large fluctuations are from Gillespie's algorithm merely taking the mean and standard variation from several simulations. Such runs easily lend themselves to the use of supercomputers, the subject of a later chapter.

The reaction probability density function is defined by

\[ p(\tau, j)d\tau = \text{ probability at time } t \text{ that the next reaction is } j, \text{ and that} \]
\[ \text{it occurs in the small time interval } (t + \tau, t + \tau + d\tau). \]
Figure 2.3: A partition of the “jump” interval. Each subinterval has equal length $\varepsilon$.

Next, one must find an expression for $p(\tau, j)d\tau$. Recall that the quantity $a_j(x)d\tau + o(d\tau)$ is the probability that reaction $j$ will occur in the interval $(t + \tau, t + \tau + d\tau)$ given $X(t) = x$. Now, $p(\tau, j)d\tau$ can be calculated as the product of two probabilities:

$$p(\tau, j)d\tau = p_0(\tau) \cdot a_j(x)d\tau.$$  \hspace{1cm} (2.19)

$p_0(\tau)$ is the probability at time $t$ that no reaction occurs in the interval $(t, t + \tau)$, while $a_j(x)d\tau$ is the probability reaction $j$ occurs in the interval $(t + \tau, t + \tau + d\tau)$.

In order to find an expression for $p_0(\tau)$, first divide the interval $(t, t + \tau)$ into $K$ subintervals of equal length $\varepsilon$, as illustrated in figure 2.3. According to a classic argument in probability, for events $A_1, A_2, ..., A_K$,

$$P\left(\bigcap_{k=1}^{K} A_k\right) = P\left(A_K \mid \bigcap_{k=1}^{K-1} A_k\right) P\left(A_{K-1} \mid \bigcap_{k=1}^{K-2} A_k\right) \cdots P(A_1).$$

Therefore, the event that no reaction occurs in subinterval $(t + (k - 1)\varepsilon, t + k\varepsilon)$ has
probability

\[ p_{0,k}(\tau) = \prod_{j=1}^{M} \left[ 1 - a_j(x) \cdot (t + k\epsilon - (t + (k - 1)\epsilon)) + o(\epsilon) \right] \]  
\[ = \prod_{j=1}^{M} \left[ 1 - a_j(x) \cdot (\epsilon) + o(\epsilon) \right] \]  
\[ = 1 - \sum_{j=1}^{M} a_j(x)\epsilon + o(\epsilon). \]  

The quantity \( p_{0,k}(\tau) \) is the probability no reaction occurs in the \( k \)th subinterval given no reaction has occurred in the previous \( k - 1 \) intervals. Thus, \( p_0(\tau) \) is the product of the individual \( p_{0,k}(\tau) \) entities. A glance at equation (2.22) reveals \( p_{0,k}(\tau) \) doesn't depend on \( k \). As a result,

\[ p_0(\tau) = \prod_{k=1}^{K} p_{0,k}(\tau) \]
\[ = [p_{0,k}(\tau)]^K \]
\[ = \left[ 1 - \sum_{j=1}^{M} a_j(x)\epsilon + o(\epsilon) \right]^K \]
\[ = \left[ 1 - \sum_{j=1}^{M} a_j(x)\tau/K + o(\tau/K) \right]^K \]
\[ = \left[ 1 - \left( \sum_{j=1}^{M} a_j(x)\tau + o(\tau) \right) / K \right]^K. \]
Finally, taking the limit as $K$ approaches infinity,

\[ p_0(\tau) = \exp(- \sum_{j=1}^{M} a_j(x)\tau). \]  \hspace{1cm} (2.23)

Plugging the expression in (2.23) into (2.19), one can finally write an expression for the reaction probability density function (pdf):

\[ p(\tau, j) = a_j(x) \exp \left( - \sum_{j=1}^{M} a_j(x)\tau \right). \]  \hspace{1cm} (2.24)

The domain for this pdf is given by $\tau \in [0, \infty)$ and $j \in \{1, \ldots, M\}$. As expected, integrating (2.24) over this set yields a value of unity.

In order for the reaction pdf to be effective, one must find a way of randomly generating the pair. Consider $p(\tau)$, the density function for the time step. According to basic probability theory, this function can be calculated by simply integrating $p(\tau, j)$ over all $j$:

\[ p(\tau) = \sum_{j=1}^{M} p(\tau, j) \]

\[ = \sum_{j=1}^{M} a_j(x) \exp \left( - \sum_{j=1}^{M} a_j(x)\tau \right) \]

\[ = a_{\text{tot}}(x)e^{-a_{\text{tot}}(x)\tau}, \]

where $a_{\text{tot}}(x) = \sum_{j=1}^{M} a_j(x)$. 
According to the probability density inversion method, in order to generate \( \tau \), then the identity \( U = F(\tau) \) must be manipulated. [12] \( F(\tau) \) is the cumulative distribution function of \( \tau \). \( U \) is a random number from the uniform distribution on the interval \([0, 1]\). Thus, to find a sample \( \tau \) from the distribution given by (2.24),

\[
U = F(\tau)
= \int_{0}^{\tau} a_{tot}(x)e^{-a_{tot}(x)\tau} dt.
\]

Solving for \( \tau \),

\[
\tau = \frac{1}{a_{tot}(x)} \ln(1 - U).
\]

Notice that \( 1 - U \) also has a uniform distribution on \([0, 1]\). For the purpose of computational efficiency, it can be replaced with \( U \).

To find a way of generating the next reaction \( j \), a similar procedure is followed. The probability density function for reaction \( j \) is computed as follows:

\[
p(j) = \int_{0}^{\infty} p(\tau, j) d\tau
= \frac{a_{j}(x)}{a_{tot}(x)}.
\]

There are two different ways to extract \( \tau \) and \( j \) from \( p(\tau, j) \). They are discussed in the next section.
2.4.2 The Algorithm

These steps give a brief summary of Gillespie's algorithm:

1. Initialize time $t=0$ and state vector $X_0(t)$. Store reaction rate vector $c = (c_1, ..., c_M)$.

2. From $X_0(t)$ and $c$, calculate and store corresponding propensity vector $a = (a_1, ..., a_M)$.

3. Using the reaction probability density function $p(t, j)$, generate which reaction $j^*$ occurs next and the respective time step $\tau_j$.

4. Update time with $t \rightarrow t + \tau_j$ and state vector with $X \rightarrow X + \nu_j$. Go back to step (2).

Gillespie [16] presents two different ways to execute step (3). One is more efficient, while the other is more descriptive and straightforward in terms of clarity. Both are described in more detail below.

2.4.2.1 The First Reaction Method

The first reaction method is easier to code and highlights subtleties involved in generating the time step. First, a "prospective" time step $\tau_j$ may be generated for
each reaction $j$. After manipulation of $p(\tau, j)$, each step is given by

$$
\tau_j = -\frac{1}{a_j(x)} \ln(U_j),
$$

where $U_j$ is a random number from the uniform distribution on $[0, 1]$. Finally, $\tau$ is selected by

$$
\tau = \min_{j=1,2,...,M} \tau_j.
$$

The next reaction that occurs is simply the $j$ corresponding to this minimum. [14] Notice the larger the value of $a_j$, the smaller the value of $\tau_j$. In turn, the smaller $\tau_j$, the more likely $j$ is to be the next reaction.

The first reaction method is introduced to add insight to the stochastic simulation model. Common intuition would lead one to base the size of the time step on how large the reaction probability $a_j(x)dt$ is. However, for nearly all systems, the following method is the more suitable algorithm to implement from an efficiency standpoint.

2.4.2.2 The Direct Method

According to the direct method, step (3) is executed using only two random numbers. First, generate $\tau$ according to

$$
\tau = -\frac{1}{a_{int}(x)} \ln(U_1), \quad (2.25)
$$
where $U_1$ is a random number from the uniform distribution on $[0, 1]$. The corresponding reaction is found by taking $j^*$ to be the integer such that

$$\sum_{j=1}^{j^*-1} a_j(x) \leq U_2 a_{tot}(x) \leq \sum_{j=1}^{j^*} a_j(x),$$

with $U_2$ being another random number from the uniform distribution on $[0, 1]$. The condition in (2.26) ensures that reaction $j$ is chosen with probability $\frac{a_j}{a_{tot}}$.

The direct method generally makes for faster simulations than the first reaction method, especially for reaction systems containing many reactions. Each loop of the first reaction method requires the generation of $M$ random numbers, while the direct method requires only two random numbers per loop. This is a substantial consideration, since generating random numbers is computationally expensive. Though different in implementation, it should be emphasized that the two methods are equivalent. [16]

### 2.4.3 Gillespie Algorithm by Example

Figure 2.4.3 compares results of stochastic simulation algorithm runs to deterministic model runs for the examples in (2.12) and (2.16)-(2.18). The overall behavior of the time trajectories are the same for each model. Still, Gillespie’s algorithm exhibits the noisy chemical fluctuations often seen in biochemical data. To the casual observer, the way the stochastic trajectories oscillate about their ODE counterparts
Figure 2.4: (Left Panel) Time dependent behavior of molecules in $A + B \rightarrow C$ reaction according to Gillespie’s algorithm (thick lines). The thin solid lines are results from the corresponding deterministic model. (Right Panel) Results from simulation of chemical reaction system described in equations (2.16)-(2.18). Again, Gillespie’s algorithm (thick lines) and the deterministic approach (thin lines) are compared.

may suggest that in some limit, the former approaches the latter. In fact, there is

truth to this conjecture. This is the subject of a later chapter.

2.5 Improvements Upon Exact Stochastic Simulation

The importance of Gillespie’s algorithm in the field of simulating chemical time

trajectories cannot be underestimated. Still, it’s hard to overlook its large computa-
tional expense. There are two main ways to speed up the exact stochastic algorithm.

One way is to find steps in the algorithm that can be exploited for efficiency. The

other way is using mathematical approximations of the underlying theory for bet-
ter results. The former still yields an exact simulation of the trajectories. Though

an improvement over strictly deterministic models, the latter isn’t as accurate as
the exact stochastic simulation algorithm.

2.5.1 Efficient Exact Simulations

Gibson and Bruck [14] proposes a more efficient implementation of the first reaction method. Called the next reaction method, this method cleverly utilizes the unused times and updates only the reaction propensities affected by the previous reaction. [8] As a result, only one random number is used per iteration as opposed to \( M \), the number of reactions.

The following is a sketch of the next reaction method's major steps:

1. Initialize molecular populations and time.
2. From stoichiometric matrices, generate dependency graph describing which reactions affect which propensities.
3. Calculate initial reaction propensities \( a_j \) and corresponding reaction times \( \tau_j \), where \( \tau_j \) has an exponential distribution with mean \( \frac{1}{a_j} \). Create a priority queue of ordered pairs \((j, \tau_j)\). Priority is given based on how small each \( \tau_i \) is.
4. The next reaction \( j^* \) is the one corresponding to the smallest time \( \tau_j^* \). Update the state \( X(t) \) to reflect this change.
5. Update the time using \( t \leftarrow \tau_j^* \).
6. Using the dependency graph, update only the reaction propensities affected by reaction \( j^* \).
(7) For all reactions \( j \neq j^* \), let \( \tau_j = \frac{a_j \text{old}(x)}{a_j \text{new}(x)} (\tau_j - t) + t \). For reaction \( j^* \), generate a new time \( \tau_j \) using the same procedure as in step (3). Amend the priority queue to reflect these changes.

(8) Repeat steps (4)-(8).

The way in which step 6 updates time may appear abnormal. However, Gibson and Bruck rigorously prove its equivalence to Gillespie's original formulation. Also observe that the next reaction method keeps track of the actual time, not time steps. This is done for efficiency purposes.

The purpose of the dependency graph is to make sure only those propensities that are affected by the previous reaction change. For instance, consider a reaction system introduced in a previous section:

\[
A \xrightarrow{k_1} B,
\]

\[
B + E \xrightarrow{k_2} BE \xrightarrow{k_4} C + E,
\]

\[
A + C \xrightarrow{k_5} 2A.
\]

Should the first reaction occur, the next reaction method would consult the dependency graph and conclude reaction propensities 1,2, and 5 are affected. Thus, there would only be 3 new calculations of the a vector as opposed to 5. The computa-
tional acceleration is even more drastic for larger, more loosely coupled systems. In general, Gibson and Bruck's method has a simulation time proportional to \( \ln(M) \), and not \( M \).

The priority queue keeps track of the possible reaction times. In computer science, this structure is better known as a simplified example of a tree. Each parent in the tree has exactly one child. The lower the time, the higher its priority in the tree.

For smaller systems, the maintenance of the two data structures dominates CPU time. For larger systems, it's widely believed the benefits of fewer calculations outweighs such costs. In examples with many loosely coupled reactions, this is the case hands down. Loosely coupled systems imply highly sparse dependency graphs and, therefore, fewer propensity updates. The next reaction method's effectiveness for systems lacking loose coupling has been called into question. [8] Though there are fewer calculations, the cost of the dependency graph and priority queue wasn't taken into account. The following discussion expands upon this deficiency and introduces a suggested improvement.

Cao, Li, and Petzold [8] introduce an alternative called the optimized direct method. As its name suggests, it's a modification of the direct method. The authors claim the next reaction method can be slower than the original direct method for systems without loose coupling and look elsewhere to improve computational efficiency. According to their experiments, the simulator spends much of its time
<table>
<thead>
<tr>
<th>Average CPU Time</th>
<th>Optimized Direct Method</th>
<th>Next Reaction Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>86.8s</td>
<td>170.4s</td>
</tr>
</tbody>
</table>

Table 2.1: Average CPU time comparison of original direct method and next reaction method on heat response shock model. Times are given in terms of seconds per simulation. Results are from Ref. [8].

maintaining the priority queue.

The optimized direct method uses ideas from Gibson and Bruck’s algorithm without the complications of extra data structures. First of all, large chemical reaction systems are usually multi-scale. In other words, a small number of reactions will do most of the firing. Recall the step in the direct method where the algorithm searches for the reaction interval in which the random number falls. Re-indexing the reactions such that the ones with the greatest $a_i$ values are first should allow for faster prediction of which reaction fires next. Table 2.1 compares simulation times for both methods for the heat shock response (HSR) model. [30, 31] The HSR system describes how *E. coli* bacteria responds to an increase in temperature. There are 28 molecular species and 61 different reactions involved. Results were taken from Ref. [8].

### 2.6 The $\tau$-Leap Algorithm

Gillespie realized the need for faster stochastic simulations and introduced the $\tau$-leap method. [19] Though similar to exact stochastic simulation, this approach simulates several reactions per larger time steps. A time step value $\tau^p$ over which
the reaction propensities don't change significantly must be determined.

Since the propensities are assumed to be relatively constant for step length \( \tau^p \), a Poisson random variable \( \mathcal{P}_j \) with mean \( a_j(x) \tau \) yields how many times each reaction fires over this interval. Thus,

\[
\mathbf{u} = \sum_{j=1}^{M} [a_j(x) \tau] \nu_j
\]

is the approximate expected change in the state vector \( \mathbf{x} \).

Now, it's desired that \( |a_j(x + \mathbf{u}) - a_j(x)| \) be small for every \( j \). Using a Taylor's expansion,

\[
|a_j(x + \mathbf{u}) - a_j(x)| \approx |\mathbf{u} \cdot \nabla a_j(x)| = \sum_{i=1}^{N} \tau \xi_i(x) \frac{\partial}{\partial x_i} a_j(x)
\]

where \( \xi_i = \sum_{j=1}^{M} a_j(x) \nu_j \). It's also desired that the change in the propensity vector is less than a fraction of the propensity sum \( a_{tot} \):

\[
\left| \sum_{i=1}^{N} \tau \xi_i(x) \frac{\partial}{\partial x_i} a_j(x) \right| \leq \varepsilon a_{tot}(x) \quad (2.27)
\]

Rearrangement of (2.27) yields

\[
\tau = \varepsilon a_{tot}(x) / \left| \sum_{i=1}^{N} \xi_i(x) \frac{\partial}{\partial x_i} a_j(x) \right|.
\quad (2.28)
\]

The leap value \( \tau^p \) should be chosen such that this value on the right hand side of
(2.28) is as small as possible:

\[
\tau^{lp} = \min_j \epsilon a_{\text{tot}}(x) / \left| \sum_{i=1}^{N} \xi_i(x) \frac{\partial}{\partial x_i} a_i(x) \right|.
\]

This formula for choosing \( \tau \) yielded good results. Nevertheless, there was still room for improvement. Better leap conditions can be found in Refs. [6] and [20].

One drawback of \( \tau \)-leaping is the possibility of negative populations. When a chemical species becomes small in number, there may be more reaction firings than molecules available. In addition, Poisson random variables can be expensive to generate. Though the chemical Langevin equation can be a suitable approximation when the propensities are large enough, simulations may benefit more from using a combination of the basic methods described here. The following chapter reviews some of these.
Chapter 3

Review of Multi-scale Methods

Some chemical systems have aspects not entirely captured by approaches that are strictly stochastic or deterministic. For example, genetic regulation often involves molecular types present in miniscule amounts (e.g. DNA) along with types of more abundance (e.g. RNA). In addition, reaction propensities may have a wide range of values. Gillespie’s exact stochastic simulation algorithm is more than sufficient in accurately modeling these systems; but its slow computational speed is a setback. How can these issues be accounted for? Hybrid models combining aspects of ordinary differential equations theory and stochastic chemical kinetics are the key to settings that straddle both sides of the fence. This chapter reviews existing hybrid methods. Table 3.1 lists them.
<table>
<thead>
<tr>
<th>Method</th>
<th>Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haseltine and Rawlings*</td>
<td>Direct SSA w/ Langevin Eqn. or ODE</td>
</tr>
<tr>
<td>Griffith et al.</td>
<td>Direct SSA, ODE</td>
</tr>
<tr>
<td>Wagner et al.</td>
<td>First Reaction SSA, Gaussian Dist., ODE</td>
</tr>
<tr>
<td>Salis and Kaznessis</td>
<td>Next Reaction SSA, Langevin Eqn.</td>
</tr>
<tr>
<td>Alfonsi et al.</td>
<td>SSA, ODE</td>
</tr>
<tr>
<td>Kierzek and Puchalka</td>
<td>Next Reaction SSA, $\tau$-leap</td>
</tr>
<tr>
<td>Harris and Clancy*</td>
<td>Next Reaction SSA, $\tau$-leap, Langevin Eqn., ODE</td>
</tr>
<tr>
<td>Burrage et al.</td>
<td>Direct SSA, $\tau$-leap, Langevin Eqn.</td>
</tr>
<tr>
<td>Turner and Cox</td>
<td>First Reaction SSA, $\tau$-leap, Langevin Eqn., ODE</td>
</tr>
</tbody>
</table>

Table 3.1: Overview of multi-scale reaction system algorithms. The "*" indicates there's no abundance threshold.

### 3.1 Scale Coupling Approaches: Two-Scale Systems

#### 3.1.1 The Haseltine and Rawlings Algorithm

Haseltine and Rawlings propose an algorithm that partitions the system with respect to reaction speed only. [24] Faster reactions are modeled using either ODE’s or the chemical Langevin equation. The direct method is used for slower reactions.

However, the direct method must be modified since the propensities are varying between firings of the slow reactions. Haseltine and Rawlings show that this may be accounted for by finding the time to the next slow reaction. This is done by solving for $\tau$ in

$$\int_{t}^{t+\tau} a_{tot}^s(X(s))ds = -\ln U_1,$$

where $a_{tot}^s(X(s))$ is the sum of the propensities of the slow reactions. Here, $t$ is either the initial time or the time of the last slow reaction. Thus, this integral must be computed as one integrates the equations that track the fast reactions. After each
slow reaction, it is reset to 0 and a new uniform random variable $U_1$ is generated.

The following outlines the major steps of the Haseltine and Rawlings algorithm:

1. Initialize time $t = 0$, propensity classification threshold, and molecular populations. Set slow reaction propensity indicator, denoted by $I$, to 0. Establish criteria for which partitioning is needed.

2. If the system does not require partitioning, perform the exact SSA for one time step, updating the system state and propensities accordingly. Repeat this procedure as long as partitioning is not needed.

3. Take two random numbers $U_1$ and $U_2$ from the uniform distribution on $[0, 1]$.

4. Using Euler’s formula, update each species involved in a fast reaction:

$$X(t + \tau) = X(t) + \nu_j a_j^f(X(t)) \tau.$$  \hspace{1cm} (3.1)

Set $I \leftarrow I + a_{tot}^s(X(t)) \tau$. Adjust propensities and sum $a_{tot}^s(X(t))$ to reflect changes in $X$ from (3.1). Repeat this step until $I$ surpasses $-\log(U_1)$.

5. Find next slow reaction by choosing $j^*$ such that

$$\sum_{j=1}^{r-1} a_j^f(x) < U_2 a_{tot}^s(x) \leq \sum_{j=1}^{r} a_j^f(x).$$

6. Update the system state and propensities to reflect this change. Return to second step.
It should be noted that the Euler update can be replaced with an Euler-Maruyama update if one desires to use the chemical Langevin equation for fast reactions:

\[ X(t + \tau) = X(t) + v_j a_j^f(X(t)) \tau + v_j \sqrt{a_j^f(X(t))} \tau N(0, 1). \]

The model is applied successfully to a couple of examples; but the issue of how to partition reactions isn’t discussed thoroughly. Also, changes in reaction classifications are not done automatically. They require user intervention. It’s also possible that reactions may remain in the same class throughout the entire simulation, regardless of how large or small their propensities become.

### 3.1.2 Dynamic Partitioning

Griffith et al. use “dynamic partitioning”. [21] Like Haseltine and Rawlings, ODE’s are used for fast reactions, while the direct method is utilized for slow ones. The set of fast reactions is defined to be the largest set such that for every reaction \( j \), \( X_i > \Lambda |v_j| \), and \( a_j \geq \beta \max(a_1, ..., a_M) \), with \( i \) being the index of any species involved in reaction \( j \). Nonnegative constants \( \Lambda \) and \( \beta \) control the classifications. The second condition, which actually permits the propensity threshold to change within the course of the simulation, is dependent on the distribution of reaction propensities.
3.1.3 Possibilities of More Than One Slow Reaction Per Time Step

Salis and Kaznessis acknowledge the possibility two slow reactions can occur within the same time step. [42] Their algorithm uses the next reaction method for slow reactions, though it’s stressed that the other versions of the exact SSA could apply as well. This is advantageous, because each slow reaction is treated separately, allowing for more than one slow reaction to occur per time step. Two different suggestions are proposed for handling this situation: (1) Either backtrack to the previous values for X(t) and use a smaller time step, or (2) approximate where each reaction occurred in the interval. A similar approach can be found in Alfonsi et al. [1, 2] Like the algorithm in Ref. [42], this approach allows for reaction reclassification during the course of the simulation. If X_i < \Lambda for some i, then any reaction j that satisfies v_{ji} \neq 0 is considered slow. Though somewhat effective, this can have negative ramifications, some of which are mentioned in section 4.1.

3.2 More Hybrid Methods: Using Leaping to Determine Time Steps

For the most part, strategies in the previous section use fixed time steps. Methods described in this section rely on "leaping" to find the next step. This allows the time step to adapt to the size of propensities in the reaction systems at hand. Some variation of the SSA (usually the next reaction implementation) is used for slow
reactions. Flexibility is key, as reclassifications of reaction speeds are preformed throughout the simulation.

3.2.1 The Maximal Time Step Method

Kierzek and Puchalka [41] introduce the maximal time step method. First, reaction partitioning occurs according to propensity values and a user-selected value $\beta$. If $a_j < \beta a_{i0}$ or all the reactants in reaction $j$ are less than another user-specified constant $\Lambda$ (or less than $\Lambda/2$ if $v_{ji} = 2$), it's labeled a slow reaction. All others are considered fast. The suggested value for $\Lambda$ is 100. A leap value $\tau$ is selected according to the $\tau$-leap method [19] applied to the fast reactions. Putative reaction times are generated for each slow reaction. If $\tau$ is smaller than the smallest putative time, the fast reactions are simulated according to the $\tau$-leap approach. However, if the smallest putative slow reaction time is smaller than $\tau$, it's accepted as the new time step. The slow reaction corresponding to this step fires and the $\tau$-leap method is applied to the fast reactions using this time step. For slow reactions, steps 6-7 of Gibson and Bruck's algorithm is used to calculate the putative slow reaction times. After each iteration, propensities, reaction classifications, and putative times are adjusted.
3.2.2 The Partitioned Leaping Algorithm

Harris and Clancy use ordinary and stochastic differential equations in addition to exact SSA and τ-leaping to simulate chemical trajectories. [22] Termed the partitioned leaping algorithm, their contribution considers propensity size as the lone factor in reaction classification. Table 3.2 gives a description of how this is carried out. Like the maximal time step method, putative reaction times are calculated for the slowest class of reactions (those using the exact SSA). If the leaping value τ is less than or equal all the putative slow times, it’s chosen as the time step. Otherwise, τ is set to the smallest putative time \( \min_j \tau_j^{nu} \), and reactions are reclassified according to the new values of \( a_j/\tau \). Of course, the reaction indexed by the smallest putative time fires. Harris and Clancy also employ the use of Gibson and Bruck’s method for updating slow reaction propensities.

As with all leaping strategies, there’s the possibility of attaining negative populations. The authors claim that with the correct partitioning, this problem should easily be avoided. Just in case, the “try again” strategy is employed. That is, if a particular molecular population falls below 0, the algorithm backs up, and uses the step value \( \tau/2 \). Should the smallest population still fall below 0, the process is repeated until the problem no longer occurs.
<table>
<thead>
<tr>
<th>Criterion</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_j \tau \leq 1$</td>
<td>Exact Stochastic</td>
</tr>
<tr>
<td>$a_j \tau &gt; 1$ but $a_j \tau \gg 1$</td>
<td>Poisson r.v.</td>
</tr>
<tr>
<td>$a_j \tau \gg 1$ but $\sqrt{a_j \tau} \gg 1$</td>
<td>SDE</td>
</tr>
<tr>
<td>$\sqrt{a_j \tau} \gg 1$</td>
<td>ODE</td>
</tr>
</tbody>
</table>

Table 3.2: Reaction propensity classification criteria for the partitioned leaping algorithm. [23]

3.3 Consideration of Molecular Abundance

The following approaches pay more attention to how the amounts of reactants and products in a reaction affect its classification.

3.3.1 The Method of Burrage, Tian, and Burrage

Burrage et al. propose a multi-scale algorithm that uses the direct method for slow reactions, $\tau$-leaping for intermediate reactions, and the chemical Langevin equation for fast reactions. [4] Classifications take into account reaction speed and molecular abundance. The slow classification is used for reactions involving species few in number, regardless of how large their propensity values are. Reactions that have large speeds and species that are all large in number are labeled fast. All other reactions are classified as intermediate. Intermediate and fast reactions are updated using $\tau$-leaping and the Euler-Maruyama method respectively, until the time point of the next slow reaction. Propensities are updated and the procedure repeats itself. Note that propensities for slow reactions aren’t updated until after a slow reaction occurs. Table 3.3 gives a description of how reactions are classified.
<table>
<thead>
<tr>
<th></th>
<th>Small</th>
<th>Moderate</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>SSA</td>
<td>SSA</td>
<td>SSA</td>
</tr>
<tr>
<td>Moderate</td>
<td>SSA</td>
<td>τ-leap</td>
<td>τ-leap</td>
</tr>
<tr>
<td>Large</td>
<td>SSA</td>
<td>τ-leap</td>
<td>Deterministic</td>
</tr>
</tbody>
</table>

Table 3.3: This table shows the algorithmic method used in the approach in Burrage et al. [4] for each combination of reaction speed (rows) and molecular abundance (columns).

Also of interest is the method of time step determination. The time step for the group of slow reactions is

\[
\tau_s = \frac{1}{a_{tot}^s} \ln(U_1),
\]

where \( a_{tot}^s \) is the sum of the slow reaction propensities and \( U_1 \) is a random number from the uniform distribution on the unit interval. For intermediate reactions, the time step is as follows:

\[
\tau_I = \min_j \left\{ \frac{a_j(X(t) + v_j) - a_j(X(t))}{a_{tot}^I(X(t))} \right\},
\]

where \( a_{tot}^I \) is the sum of the intermediate reaction propensities. The index \( j \) is only taken over intermediate reactions. If \( \tau_s < \tau_I \), \( \tau_s \) is used as the time step for both the slow and intermediate reactions. If \( \tau_I < \tau_s \), the τ-leap method is applied to intermediate reactions until \( t + \tau_s \) is reached. A slow reaction is then chosen according to the direct SSA procedure. Finally, the fast reactions are simulated using the Euler-Maruyama method from \( t \) to \( t + \tau_s \).
Burrage et al. apply their algorithm to a genetic regulation system involving lac digestion. More details on this will be given in the following chapter. For the biological example used, a set of suggested thresholds is given. Chemical species are labeled small if there are less than 100 molecules present, large if there are more than 1000, and moderate if in between. If \( a_j(X) \leq 5 \), the reaction is labeled small. Reactions with \( a_j(X) > 100 \) are labeled large. If \( 5 < a_j(X) \leq 100 \), the label is moderate.

Simulation speed is increased, but can seriously be hindered with the inclusion of reactions with large propensities under the slow category. Smaller time steps may be needed to capture adequate accuracy. Our method circumvents this issue by using \( \tau \)-leaping for such reactions. This is discussed in more detail in section 4.1.

### 3.3.2 The COAST Algorithm

The Controllable Approximative Stochastic Reaction Algorithm (COAST) is introduced in Ref. [47]. The authors argue that reactions involving intermediate and large numbers of molecules can be simulated by determining the time spans for which the counts of their associated species are nearly constant. Under this condition, reaction firings are considered nearly independent processes. For each reaction, \( l_j \), the critical number of reactions before an appreciable change in molec-
ular number, is calculated:

\[ l_j = \min_{i: \alpha_{ji} > 0, \nu_{ji} > 0} \frac{eX_i - 2\eta_j v_{ji}}{\epsilon} \]

where \( \eta_j \) is the total number of reactions where either \( \alpha_{ji} > 0 \) or \( \kappa_{ji} > 0 \) and \( \epsilon \) is a parameter that controls accuracy. Approximate mean times \( \tau_j \) are then calculated for each \( l_j \). The smallest time \( \tau = \min_j \tau_j \) is used as the time step.

Next, reaction probabilities \( p_j \) are calculated. By definition,

\[ p_j = l_j \tau / \alpha_{ji}^* \]

where \( \alpha_{ji}^* = \min_i \alpha_{ji} : \alpha_{ji} > 0 \). Much like propensities, this expression gives a sense of how likely (or how many times) a reaction is to fire. Each reaction \( j \) is categorized based on the size of \( p_j \).

The first category, composed of reactions satisfying \( \alpha_{ji}^* \leq 1/[3\alpha p_j (1 - p_j)] \), uses the modified first reaction method. It contains reactions involving species all low in number and typically low in speed. The SSA for individual reactions is used to simulate reactions in this class.

The second category involves reactions obeying the constraint

\[ 1/[3\alpha p_j (1 - p_j)] < \alpha_{ji}^* \leq (1 - p_j)/(p_j \alpha^3). \]  

(3.2)
This model is termed the approximative stochastic method. For probabilities satisfying \((3.2)\), the number of times a reaction fires can be described by discrete Gaussian distributions. The update rule can be likened to a Gaussian distribution rounded to the nearest integer:

\[
X_i \rightarrow X_i + \nu_j \text{round}(\alpha_j^* p_j + \sqrt{\alpha_j^* p_j (1 - p_j)} N(0, 1)).
\]

\(N(0, 1)\) is a random number from the standard normal distribution. Should the rounded value be negative, there is no contribution from reaction \(j\), as the update value is zero. If the rounded value results in a negative value for \(X_i\), then \(X_i\) is simply reset to 0.

The final category concerns even larger \(p_j\) values. For reactions satisfying

\[
\alpha_j^* > (1 - p_j)/(p_j \alpha^3),
\]

the stochastic term in \((3.3)\) is negligible. Therefore, the update rule is given by

\[
X_i = X_i + \nu_j \text{round}(\alpha_j^* p_j).
\]

This model, called the deterministic reaction kinetics method, is just the rounded form of the solution to the aforementioned ODE models.
Chapter 4

The Adaptive Multi-scale Simulation Algorithm

4.1 The Adaptive Multi-scale Simulation Algorithm

A new method will now be presented for simulating the molecular counts of a reaction system. The Adaptive Multi-scale Simulation Algorithm (AMSA) uses a mix of stochastic and ordinary differential equations and discrete simulation. For most iterations, a fixed time step $\tau$ is utilized. It is occasionally necessary to use shorter time steps to avoid negative populations from $\tau$-leap updates. Details are given below. Like the algorithm in Burrage et al. [4], reaction partitioning depends on both reaction speed and molecular abundance. There are three different classifications for the propensities: slow, medium, and fast. Molecular abundance
can be placed in one of two different categories: the macroscale class for species large in number and microscale class for species present in smaller numbers. When classifying reactions in terms of molecular abundance, we consider only those chemical species whose molecular number is changed by the reaction (i.e., $v_{ji} \neq 0$). Reactions involving only macroscale species are considered macroscale. If they have any microscale species, they’re considered microscale. ODE’s are used to model fast reactions with all macroscale species, while the chemical Langevin equation models medium reactions with strictly macroscale species. A variant of the first reaction method is used for slow reactions, regardless of the scale. Table 4.1 shows the type of algorithm used for each combination of propensity speed and molecular scale.

The reaction types not mentioned above have medium and fast propensities with a microscale species. Ref. [4] categorizes these as slow. Treating such reactions with the SSA can significantly slow simulations. Larger propensities modeled per reaction means smaller time steps. Differential equations models aren’t an ideal fit either, since it’s desired that microscale species be assigned integer quantities. As a result, a Poisson random variable with mean $a_j \tau$ (similar to the $\tau$-leap method) is the better choice. This allows several reactions to be simulated per time step while keeping microscale populations as integers.

There’s an ample choice of numerical methods that can be utilized to determine the differential equations updates. For purposes of ease, Euler’s method is used
<table>
<thead>
<tr>
<th>Microscale</th>
<th>Macroscale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow</td>
<td>First Reaction</td>
</tr>
<tr>
<td>Medium</td>
<td>Poisson r.v.</td>
</tr>
<tr>
<td>Fast</td>
<td>First Reaction Chemical</td>
</tr>
<tr>
<td></td>
<td>Langevin ODE</td>
</tr>
</tbody>
</table>

Table 4.1: This table shows the algorithmic method used in AMSA for each combination of reaction speed (rows) and molecular abundance (columns).

here for the ODE's, while the chemical Langevin equation uses the Euler-Maruyama formula. [25] Suggestions for alternate numerical methods are given in Ref. [4].

### 4.1.1 A Discussion of Thresholds

The accuracy and speed of AMSA can be mainly adjusted in two ways: time step and classification thresholds. Larger time steps and lower thresholds result in faster simulations, but at the price of decreased accuracy in comparison with the exact SSA. Time step selection depends highly on the system at hand, and thus, will be discussed in the examples section. For the present analysis, the lower threshold for reaction speed $\Gamma_1$ is chosen such that slow reactions have at most a 10 percent chance of occurring. In other words, all reactions $j$ such that $a_1/\tau < .1$ are classified as slow. There is obviously some error associated with this choice, as a particular slow reaction could fire twice in an interval of length $\tau$. There is a $(.1)^2 = .01$ chance this will be the case, a relatively low probability. One could incorporate suggestions from Ref. [42] for dealing with this issue, but we have not done so at this point.
The upper threshold $\Gamma_2$ distinguishes between medium and fast propensities. For reactions with strictly macroscale species, it is utilized to decide whether to use the chemical Langevin equation or Euler's method for solving ODE's. Recall the update rule for a macroscale medium reaction $j$:

$$X(t + \tau) = X(t) + \nu_j a_j(X(t))\tau + \nu_j \sqrt{a_j(X(t))\tau} N(0, 1).$$

The last term on the right hand side gives the size of the statistical fluctuations in the reaction. Because $\lim_{\tau \to \infty} \frac{\sqrt{\tau}}{\tau} = \frac{1}{\sqrt{\tau}} = 0$, this term should become small relative to the deterministic term, for large enough values of $a_j(X(t + \tau))$. In other words, it's desired that

$$\left(a_j(X(t + \tau))\tau\right)^{-1/2} < \Gamma_2^{-1/2},$$

for some large value of $\Gamma_2$. Both examples in this chapter use $\Gamma_2 = 177.78$. Thus, ODE's are used when statistical fluctuations are less than 7.5 percent of the value of $a_j\tau$.

Molecular populations below abundance threshold $\Lambda$ are treated as discrete. We use the threshold is $\Lambda = 10$ in the simulations described below. Since AMSA allows regime switching from differential equations to discrete methods, another subtle dilemma mentioned below arises. If a macroscale molecular type is reclassified as microscale, care must be taken to ensure it's rounded to the nearest integer.
4.1.2 On Negative Populations

The inclusion of Poisson random variables for faster reactions introduces yet another issue— the possibility of more reactions occurring than are allowed by the state $X(t)$. Since we are using a $\tau$-leap type algorithm, negative populations are a possibility. Other authors suggest several ways to deal with this issue. Tian and Burrage argue that a binomial approximation is suitable. [44] Ref. [5] proposes a modified $\tau$-leaping procedure. Here, a much simpler approach is adopted. Suppose that at time $t + \tau$, negative populations occur in some of the medium and fast microscale reactions. For these reactions, AMSA backs up to the last time point $t$ and linearly interpolates between $X(t)$ and $X(t + \tau)$. Call the linear interpolant $\tilde{X}_i(t + s)$ for $0 \leq s \leq \tau$. Then,

$$\tau^* = \inf \left\{ s : \min_i \tilde{X}_i(t + s) < 0 \right\}.$$

The time step is reset to be $\rho \tau^*$ for some preselected $\rho$ such that $0 < \rho < 1$. We use $\rho = .5$. Reaction speeds are reclassified. The algorithm simulates the next state using the new step $\rho \tau^*$ once, then reverts back to the original time step $\tau$. Should at least one of the populations remain negative after using the time step value $\rho \tau^*$, the process is repeated until all of the populations are nonnegative. If a negative population arises from the chemical Langevin or ODE updates, the time step is probably too large.
4.1.3 An Algorithmic Description

Listed below are the steps of AMSA.

(1) Initialize time \( t = 0 \), time step \( \tau \), molecular populations \( X(0) \), threshold scale classifications, and an exponential random variable \( -\log(U_j) \) for each slow reaction. Set slow reaction propensity sums, denoted \( I_j \), to 0.

(2) Calculate each reaction propensity \( a_j(X(t)) \) \begin{itemize}
  \item IF \( a_j(X(t + \tau)) < \Gamma_1 \), reaction \( j \) is classified as slow.
  \item ELSE IF \( a_j(X(t + \tau)) < \Gamma_2 \) AND at least one species is microscale, reaction \( j \) is classified as microscale medium.
  \item ELSE IF \( a_j(X(t + \tau)) < \Gamma_2 \) AND all species are macroscale, reaction \( j \) is classified as macroscale medium.
  \item ELSE IF \( a_j(X(t + \tau)) \geq \Gamma_2 \) and at least one species is microscale, reaction \( j \) is classified as microscale fast.
  \item ELSE reaction \( j \) is classified as macroscale fast.
\end{itemize}

END IF

(3) For each medium and fast reaction \( j \) involving a microscale species, generate a Poisson random variable \( \mathcal{P}_j \) with mean \( a_j(X(t))\tau \) to determine how many times the reaction occurs. Update the state vector: \( X(t + \tau) = X(t) + \mathcal{P}_j \nu_j \).

(4) For each medium reaction \( j \) involving macroscale species, integrate using the
Euler-Maruyama update rule: \( X(t + \tau) = X(t) + \nu_j a_j(X(t)) \tau + \nu_j \sqrt{a_j(X(t)) \tau} N(0, 1) \).

\( N(0, 1) \) is a random number from the standard normal distribution.

(5) For each fast reaction \( j \) involving strictly macroscale species, apply the Euler update rule: \( X(t + \tau) = X(t) + \nu_j a_j(X(t)) \tau \).

(6) For each slow reaction \( j \), set \( I_j \leftarrow I_j + a_j(X(t)) \tau \). If \( I_j \) exceeds \(-\log(U_j)\), set \( X(t + \tau) = X(t) + \nu_j \). Generate another exponential random variable \(-\log(U_j)\). Reset propensity sum \( I_j \) to 0.

(7) Reset \( t \leftarrow t + \tau \).

(8) Repeat steps (2)-(8) until \( t \) has surpassed a desired final time \( T \).

### 4.1.4 Uncoupling Slow Reactions

A subtle dilemma arises when applying hybrid methods that allows for reclassification. A particular reaction could cross the slow reaction threshold and become a non-slow reaction. This is problematic, considering all slow reaction propensities are integrated to determine the next slow step. One way around this is to treat each reaction as a separate entity. In AMSA, a random number is generated for each slow reaction. This section gives a proof to why this alternative is accurate.

The time until the next \( j \) reaction is found by integrating the entity \( a_j \) until it exceeds the variable. The justification for this procedure is formalized in Lemma 4.1.1; but first, a discussion on approximations is needed. The integral representation for
the state vector $\mathbf{X}$ is given by

$$
\mathbf{X}(t) = \mathbf{X}(t_0) + \sum_j v_j Y_j \left( \int_{t_0}^t a_j(\mathbf{X}(s)) ds \right).
$$

$Y_j$ can be approximated as follows:

1. For reactions modeled by the $\tau$-leap algorithm, the $a_j(\mathbf{X}(s))$ is replaced by a constant. Each constant is computed at the beginning of the time step.

2. For macroscale intermediate reactions, $Y_j(u)$ is replaced by $W_j(u) + u$, where the $W_j(u)$'s are independent Brownian motions. The integral in the argument is unchanged.

3. For fast macroscale reactions, $Y_j(u)$ is replaced by $a_j(\mathbf{X}(s))$. By the Fundamental Theorem of Calculus, these can be turned into ODE's.

More on these approximations and associated limit theorems can be found in Ref. [33]. Now, the main result is stated and proven.

**Lemma 4.1.1** If $j$ is classified as slow and remains so in the interval $[t_0, t]$, $|\mathbf{X}_i(t)| < \infty$ for all $i$ and $t \in [0, \infty)$, and there are finitely many reactions in any finite interval in $[0, \infty)$, then

$$
P(\tau_j > t | \mathbf{X}(t_0)) = \exp \left( - \int_{t_0}^t a_j(\mathbf{X}(s)) ds \right)
= P \left( Z > \int_{t_0}^t a_j(\mathbf{X}(s)) ds \right), s \in (t_0, t),
$$
where $Z$ is exponentially distributed with mean 1.

**Proof**: Observe the relation

$$P(\tau_j > t | X(t_0)) = p_0(t), \quad (4.1)$$

where $p_0(t) = P(\text{no } j \text{ reaction in } [t_0, t])$. Let $\psi_p$ be a sequence of partitions such that for every $p$, all intervals are of equal length $\Delta_p$. Index each interval for a mesh $\psi_p$ by $t_0 = s_{1,p}, ..., s_{k,p}, ..., s_{K_p,p} = t$. Now,

$$p_0(t) = \prod_{k=1}^{K_p-1} \left[ 1 - (s_{k+1,p} - s_{k,p}) a_j(X(s_{k,p})) + o(\Delta_p) \right] = \exp \left( \ln \prod_{k=1}^{K_p-1} \left[ 1 - \Delta_p a_j(X(s_{k,p})) + o(\Delta_p) \right] \right) \quad (4.2)$$

$$= \exp \left( \sum_{k=1}^{K_p-1} \ln \left[ 1 - \Delta_p a_j(X(s_{k,p})) + o(\Delta_p) \right] \right). \quad (4.3)$$

Exploiting the relation $\log(1 + x + o(x)) = x + o(x)$, the expression in (4.3) is now

$$p_0(t) = \exp \left( \sum_{k=1}^{K_p-1} \left[ -\Delta_p a_j(X(s_{k,p})) + o(\Delta_p) \right] \right)$$

Note that with the exception of the $o(\Delta_p)$ terms, the summation is a Riemann sum which converges to an integral for increasingly finer meshes. Taking the limit of
the left hand side as $\Delta_p \to 0$,

$$p_0(t) = \exp \left( - \int_{t_0}^{t} a_j(X(s)) ds \right).$$

The second equality is easily verified by substituting $Z = -\ln U$, with $U$ being a random number from the uniform distribution on $[0, 1]$. 

### 4.1.5 Important Considerations

The accuracy and speed of AMSA can be mainly adjusted in two ways: time step and classification thresholds. Larger time steps and lower thresholds result in faster simulations, but at the price of decreased accuracy in comparison with the exact SSA. Time step selection depends highly on the system at hand, and thus, will be discussed in the examples section. For the present analysis, the lower threshold for reaction speed $\Gamma_1$ is chosen such that slow reactions have at most a 10 percent chance of occurring. In other words, all reactions $j$ such that $a_j \tau < .1$ are classified as slow. There is obviously some error associated with this choice, as a particular slow reaction could fire twice in an interval of length $\tau$. There is a $(.1)^2 = .01$ chance this will be the case, a relatively low probability. One could incorporate suggestions from Ref. [42] for dealing with this issue, but we have not done so at this point.

The upper threshold $\Gamma_2$ distinguishes between medium and fast propensities.
For reactions with strictly macroscale species, it is utilized to decide whether to use the chemical Langevin equation or Euler's method for solving ODE's. Recall the update rule for a macroscale medium reaction $j$:

$$X(t + \tau) = X(t) + v_j a_j(X(t)) \tau + \nu_j \sqrt{a_j(X(t)) \tau} N(0, 1).$$

The last term on the right hand side gives the size of the statistical fluctuations in the reaction. Because $\lim_{x \to \infty} \frac{\sqrt{x}}{x} = \frac{1}{\sqrt{x}} = 0$, this term should become small relative to the deterministic term, for large enough values of $a_j(X(t + \tau))$. In other words, it's desired that

$$\left\{a_j(X(t + \tau)) \tau\right\}^{-1/2} < \Gamma_2^{-1/2},$$

for some large value of $\Gamma_2$. Both examples in this chapter use $\Gamma_2 = 177.78$. Thus, ODE's are used when statistical fluctuations are less than 7.5 percent of the value of $a_j \tau$.

Molecular populations below abundance threshold $\Lambda$ are treated as discrete. We use the threshold is $\Lambda = 10$ in the simulations described below. Since AMSA allows regime switching from differential equations to discrete methods, another subtle dilemma mentioned below arises. If a macroscale molecular type is reclassified as microscale, care must be taken to ensure it's rounded to the nearest integer.
4.2 AMSA vs. Similar Multi-scale Algorithms

Algorithms described in the last chapter signify marked improvement in the computational efficiency of reaction simulations. There still is room for improvement, though. A new method will be introduced in the following section. However, before going into more detail, distinctions should be drawn between the new approach and methods in the previous chapter that bear close resemblance.

The performance of the method in Burrage et al. can suffer if a reaction has a large propensity value and a species small in number. As an example, consider the reaction $A + B \xrightarrow{c} D$, where $X_A = 1$ and $c$ is so large that the reaction is classified as fast. The SSA would be applied. As a result, the time step needed to capture fast reaction dynamics would be so small, that simulations would be severely slowed. In addition, for every slow reaction time step $\tau_s$, fast reactions are simulated separately from others. It should be stressed that slow reaction propensities can affect fast ones and vice versa. As a result, the feedback from a drastic change in a fast reaction propensity would not be reflected in a slow reaction propensity until after $t + \tau_s$. Accuracy issues could arise, as a result. Methods like that of Haseltine and Rawlings take this into account.

The effectiveness of the COAST algorithm is limited because of similar issues. For each time step, several slow reactions can fire without feedback from larger reaction propensities. In addition, dynamics of a slow reaction can also affect faster reactions, as evidenced in the toy model of genetic regulation introduced
in the next section. There’s the possibility of accuracy problems, since negative populations are merely reset to 0. Numerical error also can arise from rounding all populations to the nearest integer.

The partitioned leaping algorithm rounds to the nearest integer to avoid fractional values for counts small in number. Our algorithm uses the same four basic methods; however, the partitioning criteria takes into account molecular abundance. As a result, two different reactions with similar propensities could use different methods depending on the scale of the molecular species. The partitioned leaping algorithm gives guidelines for choosing propensity thresholds, but these are arbitrary at best. Our algorithm uses fixed time steps. Though this limits flexibility for systems with largely varying propensities, it allows for a more definitive choice of thresholds.

4.3 Applications

Biological systems often involve molecular types of varying abundance as well as reactions that occur on widely differing time scales. Therefore, they provide a natural setting for applying multi-scale algorithms. The following sections are devoted to applying AMSA to such examples.
4.3.1 A Simple Example: Genetic Self-Regulation

AMSA is first implemented on a hypothetical reaction system that includes the basic features of genetic regulation. The first reaction involves the attachment of protein $P$ to gene $G$. Once this occurs, $G$ assumes an inactivated state $G \cdot P$:

$$G + P \xrightarrow{c_1} G \cdot P$$

When the $P$ detaches itself, the gene is free to produce transcript, denoted by $R$. In turn, $R$ catalyzes the formation of $P$:

$$G \xrightarrow{c_2} G + R$$

$$R \xrightarrow{c_3} R + P$$

Both $R$ and $P$ are degradable:

$$R \xrightarrow{c_4} \emptyset$$

$$P \xrightarrow{c_5} \emptyset.$$ 

This toy model, from henceforth termed the self-regulating gene, is insightful because it includes the basic features of genetic regulatory systems. Also, the dynamics of the negative feedback loop ($P$'s inhibition of $G$) aren't fully captured by a strictly deterministic system. As a result, the stochastic element of the hybrid and
SSA models offers a clear advantage. Similar examples appear in Refs. ([34]),([28]), and ([7]).

The time step $\tau$ is 0.0075. The total amount of bound and unbounded $G$ operators is 2. Though the numbers of $R$ and $P$ molecules aren’t given artificial bounds, they tend to have specific ranges. As mentioned earlier, $\Lambda = 10$. Of course, $G$ always remains below the lower threshold. $R$ and $P$ are classified by the macroscale, except for periods where the system is at “rest”. A “burn-in” time of length 1000 is used to allow the system to reach steady state values. The volume $V = 1$.

4.3.1.1 Timing Results

For illustration purposes, figure 4.1 shows time evolution results according to the SSA. Initial values and rate constants can be found in tables 4.2 and 4.3, respectively. The dynamics of the system are highly irregular. Unattached genes result in rapid production of $P$. When $P$’s population becomes sufficiently large, one or both genes are deactivated, limiting $P$ production. The waiting times for changes in the amount of $G$, are random. AMSA results for $P$ are compared to that of a run using a nearly optimized version of the direct SSA that at each step calculates only the propensities affected by the last occurring reaction (figure 4.2).

Table 4.6 shows the CPU time needed to simulate the system for the two methods. The simulations for the self-regulating gene were run on a 2 X 450MHz UltraSPARC-II processor by Sun Microsystems. As suggested by the results, AMSA
Figure 4.1: Time courses for self-regulation model. The exact SSA is used here.

<table>
<thead>
<tr>
<th>Reaction</th>
<th>Rate Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G + P \rightarrow G \cdot P$</td>
<td>$c_1$</td>
<td>$0.0001 , \text{(molec)}^{-1}\text{s}^{-1}$</td>
</tr>
<tr>
<td>$G \cdot P \rightarrow G + P$</td>
<td>$c_{-1}$</td>
<td>$0.075 , \text{s}^{-1}$</td>
</tr>
<tr>
<td>$G \rightarrow G + R$</td>
<td>$c_2$</td>
<td>$20 , \text{s}^{-1}$</td>
</tr>
<tr>
<td>$R \rightarrow R + P$</td>
<td>$c_3$</td>
<td>$50 , \text{s}^{-1}$</td>
</tr>
<tr>
<td>$P \rightarrow \emptyset$</td>
<td>$c_4$</td>
<td>$5 , \text{s}^{-1}$</td>
</tr>
<tr>
<td>$R \rightarrow \emptyset$</td>
<td>$c_5$</td>
<td>$0.5 , \text{s}^{-1}$</td>
</tr>
</tbody>
</table>

Table 4.2: Parameters for toy model simulation.

effectively speeds up chemical reaction system simulations while maintaining a considerable amount of accuracy.

4.3.1.2 Summary Statistics

We use certain summary statistics to study the accuracy of multi-scale simulations. They can shed more light on how close AMSA results are to those of the exact stochastic simulation algorithm. This section analyzes information about the histograms.
<table>
<thead>
<tr>
<th>Molecular Type</th>
<th>Initial Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td>1</td>
</tr>
<tr>
<td>$G \cdot P$</td>
<td>1</td>
</tr>
<tr>
<td>$R$</td>
<td>10</td>
</tr>
<tr>
<td>$P$</td>
<td>300</td>
</tr>
</tbody>
</table>

Table 4.3: Initial values for toy model simulation.

Figure 4.2: Time course for $P$ according to SSA (top) and AMSA (bottom).
Figure 4.3: Bar plots illustrating the relative amount of time spent in each possible DNA state.

Figure 4.3 shows scaled histograms for the amount of time spent in each possible gene state. For both methods, there is exactly one activated gene for about half the simulation. Figure 4.4 contains smoothed histograms for $R$. From henceforth, all histograms use a bin width of 1 because chemical trajectories simulated by the exact SSA can only integer values. There generally is good agreement between AMSA and the SSA.

As a further test of accuracy, 95 percent confidence intervals for the heights of the histograms were calculated for the $R$ distributions (figure 4.6). To compute these intervals, a long run ($T = 20000$) was divided into 20 shorter runs of equal length. Again, the graphs suggest good accuracy for AMSA.

The histogram distance, the $L_1$ norm of the difference between two histograms, can be used as a measure of how well approximate methods mimic the SSA. [9] Table
Figure 4.4: Histograms of $R$ population for SSA (solid) vs. AMSA (dashed) for $\Delta t = 0.0075$ (left) and $\Delta t = 0.1$.

Figure 4.5: Histograms of $P$ population for SSA (solid) vs. AMSA (dashed) for $\Delta = 0.0075$ (left) and $\Delta t = 0.1$ (right).
Figure 4.6: Confidence intervals for histograms of $R$ distribution for SSA (solid) vs. AMSA (dashed) for $\Delta t = 0.0075$ (left) and $\Delta t = 0.1$ (right).

Figure 4.7: Confidence intervals for histograms of $P$ distribution for SSA (solid) vs. AMSA (dashed) for $\Delta t = 0.0075$ (left) and $\Delta t = 0.1$ (right).
<table>
<thead>
<tr>
<th>Step Size</th>
<th>$R$</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.0417</td>
<td>0.0885</td>
</tr>
<tr>
<td>0.0075</td>
<td>0.0376</td>
<td>0.0376</td>
</tr>
</tbody>
</table>

Table 4.4: Histogram distance between AMSA and SSA for $R$ and $P$ distributions.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Self-Regulating Gene</th>
<th>$lac$ System</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau$</td>
<td>0.0075</td>
<td>0.0001</td>
</tr>
<tr>
<td>$\Gamma_1$</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>$\Gamma_2$</td>
<td>177.77</td>
<td>177.77</td>
</tr>
<tr>
<td>$\Lambda$</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$\rho$</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 4.5: Parameters for reaction system examples.

4.4 lists the distances between AMSA and the SSA for the $R$ and $P$ distributions. Two different time steps are utilized.

### 4.3.2 A More Complicated Example: Detailed Genetic Regulation

Kierzek examines genetic regulation in the $lac$ system. [29] *E. Coli* normally metabolizes glucose as its primary energy source. In the absence of glucose, though, lactose is used as an alternative. The $lac$ operon is a regulated locus of genes that allows for the digestion of lactose. It operates similarly to the smaller model above. When activated, thousands of lactose digesting enzyme molecules are produced.

Though Kierzek’s system doesn’t include the negative feedback loop, it does include the more sophisticated mechanisms such as mRNA synthesis and protein

<table>
<thead>
<tr>
<th>Method</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSA</td>
<td>49.19 min</td>
</tr>
<tr>
<td>AMSA</td>
<td>3.87 min</td>
</tr>
</tbody>
</table>

Table 4.6: Analysis of run times for SSA and Adaptive Multi-scale models.
chain elongation. Tables 4.8 and 4.9 list the full reaction system and the initial conditions. Table 4.9 lists the corresponding initial conditions.

The time step $\tau$ is $10^{-4}$, an arbitrarily chosen value that gave good results. The simulation spans 10 generations, each one with a duration of 35 minutes. Cell division takes place after every new generation; therefore, genetic material is doubled, and, subsequently, all populations are halved. The volume is initiated at 1, grows linearly with time, and reaches 2 at the end of the generation. It’s also halved after cell division. For this example, rate constants of reactions with two or more reactants depend on volume. Consequently, such constants must be adjusted after every step. More details can be found in Ref. [29].

4.3.2.1 Results

Figure 4.8 compares time evolution results for AMSA with those of the nearly optimized direct SSA. In particular, the trajectories shown are for “product” molecules, or lactose molecules ingested into the bacterium. Simulations were run on an Intel Pentium processor with 2.80 Ghz. Time course results for the self-regulation system can’t be superimposed upon one another due to the randomness of the trajectories. In addition, operator fluctuations aren’t rapid enough to produce deterministic-like behavior in protein that allows for superimposition. A computational timing comparison can be found in table 4.7.
Figure 4.8: Time course for product molecules for $lac$ regulation system according to one run each of the SSA (solid) and AMSA (dashed). This plot represents ten generations.

<table>
<thead>
<tr>
<th>Method</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSA</td>
<td>60.14 hr</td>
</tr>
<tr>
<td>AMSA</td>
<td>1.57 hr</td>
</tr>
</tbody>
</table>

Table 4.7: Analysis of run times for SSA and Adaptive Multi-scale models for Kierzek's lactose regulation system.
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Rate Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Plac + RNAP \rightarrow Plac \cdot RNAP$</td>
<td>$c_1$</td>
<td>0.17 (molec s)$^{-1}$</td>
</tr>
<tr>
<td>$Plac \cdot RNAP \rightarrow Plac + RNAP$</td>
<td>$c_2$</td>
<td>10 s$^{-1}$</td>
</tr>
<tr>
<td>$Plac \cdot RNAP \rightarrow TrLacZ1$</td>
<td>$c_3$</td>
<td>1 s$^{-1}$</td>
</tr>
<tr>
<td>$TrLacZ1 \rightarrow RbsLacZ + Plac + TrLacZ2$</td>
<td>$c_4$</td>
<td>1 s$^{-1}$</td>
</tr>
<tr>
<td>$TrLacZ2 \rightarrow TrLacY1$</td>
<td>$c_5$</td>
<td>0.015 s$^{-1}$</td>
</tr>
<tr>
<td>$TrLacY1 \rightarrow RbsLacY + TrLacY2$</td>
<td>$c_6$</td>
<td>1 s$^{-1}$</td>
</tr>
<tr>
<td>$TrlacY2 \rightarrow RNAP$</td>
<td>$c_7$</td>
<td>0.36 s$^{-1}$</td>
</tr>
<tr>
<td>Ribosome + RbsLacZ $\rightarrow$ Ribosome $\cdot$ RbsLacZ</td>
<td>$c_8$</td>
<td>0.17 (molec s)$^{-1}$</td>
</tr>
<tr>
<td>Ribosome + RbsLacY $\rightarrow$ Ribosome $\cdot$ RbsLacY</td>
<td>$c_9$</td>
<td>0.17 (molec s)$^{-1}$</td>
</tr>
<tr>
<td>Ribosome $\cdot$ RbsLacZ $\rightarrow$ Ribosome + RbsLacZ</td>
<td>$c_{10}$</td>
<td>0.45 s$^{-1}$</td>
</tr>
<tr>
<td>Ribosome $\cdot$ RbsLacY $\rightarrow$ Ribosome + RbsLacY</td>
<td>$c_{11}$</td>
<td>0.45 s$^{-1}$</td>
</tr>
<tr>
<td>Ribosome $\cdot$ RbsLacZ $\rightarrow$ TrRbsLacZ + RbsLacZ</td>
<td>$c_{12}$</td>
<td>0.4 s$^{-1}$</td>
</tr>
<tr>
<td>Ribosome $\cdot$ RbsLacY $\rightarrow$ TrRbsLacY + RbsLacY</td>
<td>$c_{13}$</td>
<td>0.4 s$^{-1}$</td>
</tr>
<tr>
<td>TrRbsLacZ $\rightarrow$ LacZ</td>
<td>$c_{14}$</td>
<td>0.015 s$^{-1}$</td>
</tr>
<tr>
<td>TrRbsLacY $\rightarrow$ LacY</td>
<td>$c_{15}$</td>
<td>0.036 s$^{-1}$</td>
</tr>
<tr>
<td>LacZ $\rightarrow \emptyset$</td>
<td>$c_{16}$</td>
<td>6.42e-5 s$^{-1}$</td>
</tr>
<tr>
<td>LacY $\rightarrow \emptyset$</td>
<td>$c_{17}$</td>
<td>6.42e-5 s$^{-1}$</td>
</tr>
<tr>
<td>RbsLacZ $\rightarrow \emptyset$</td>
<td>$c_{18}$</td>
<td>0.3 s$^{-1}$</td>
</tr>
<tr>
<td>RbsLacY $\rightarrow \emptyset$</td>
<td>$c_{19}$</td>
<td>0.3 s$^{-1}$</td>
</tr>
<tr>
<td>LacZ + lactose $\rightarrow$ LacZ $\cdot$ lactose</td>
<td>$c_{20}$</td>
<td>9.52e-5 (molec s)$^{-1}$</td>
</tr>
<tr>
<td>LacZ $\cdot$ lactose $\rightarrow$ LacZ + product</td>
<td>$c_{21}$</td>
<td>431 s$^{-1}$</td>
</tr>
<tr>
<td>LacY $\rightarrow$ lactose + LacY</td>
<td>$c_{22}$</td>
<td>14 s$^{-1}$</td>
</tr>
</tbody>
</table>

Table 4.8: Reactions and parameters for lac system. The symbol $\emptyset$ denotes a degradation reaction.

<table>
<thead>
<tr>
<th>Molecular Type</th>
<th>Initial Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plac</td>
<td>1</td>
</tr>
<tr>
<td>RNAP</td>
<td>35</td>
</tr>
<tr>
<td>Ribosome</td>
<td>350</td>
</tr>
<tr>
<td>all other species</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.9: Initial values for Kierzek model.
Chapter 5

Large Scale Simulations Using Supercomputers

The rise of supercomputers comes just at the right time for stochastic simulation. These powerful machines are advantageous, because they allow many runs of the same system, greatly reducing simulation time. Large clustered computer systems use programs like MPI to distribute information among the different processors.

Models that use Monte Carlo simulation, including Gillespie’s algorithm and related hybrid models, easily lend themselves to multi-node programming. One run of a stochastic simulation algorithm only gives one possible time trajectory. However several runs of the same program can greatly improve the quality of the statistical analysis.

This chapter summarizes the results from applying the AMSA to genetic regul
lation examples using hundreds of runs on supercomputer clusters. In particular, Rice University’s clusters RTC and ADA are utilized. While speed and computational efficiency are emphasized, the main focus of the chapter is statistical analysis.

5.1 Revisiting Genetic Regulation

In stochastic simulations, there are two ways to generate enough data for statistical analysis:

- run the system for a long time and split into smaller runs

- take many independent runs

The toy model in chapter 4 uses the former. For ideal situations, this is sufficient. However, the smaller divided runs aren’t entirely independent. Also, longer runs mean increased simulation time. Moreover, splitting long runs only practical for equilibrium calculations. For systems like Kierzek’s lac model, this would be inaccurate. Supercomputer clusters allow many different independent runs of the same system, at a lower computational cost. The speedup is on the order of the number of nodes in the cluster.

In the following sections, the genetic regulation systems in chapter 4 are analyzed using simulations from ADA and RTC. First, results from the self-regulation gene are reanalyzed. Then, a new reaction system is introduced as another example of hybrid methods.
5.1.1 The Self-Regulation Gene

The following are the six reactions involved in genetic self-regulation:

\[ G + P \xrightarrow{c_1} G \cdot P \]

When the \( P \) detaches itself, the gene is free to produce transcript, denoted by \( R \). In turn, \( R \) catalyzes the formation of \( P \):

\[ G \xrightarrow{c_2} G + R \]

\[ R \xrightarrow{c_1} R + P \]

Both \( R \) and \( P \) are degradable:

\[ R \xrightarrow{c_4} \emptyset \]

\[ P \xrightarrow{c_5} \emptyset. \]

Figures 5.1.1.1-5.5 give molecular distribution graphs for \( G \), \( R \), and \( P \) that correspond with figures 4.3-4.7 in the previous chapter. Runs were performed on 100 dual core 2.2 GHz AMD Opteron 275 CPUs on Rice’s ADA cluster. The first three sets of figures are virtually identical with their serial processor counterparts. However, the confidence intervals for the last two are smaller for the multi-processor
runs. Recall that the first set of graphs split a long trial into 20 smaller ones. The second set used 100 long simulations. The consequence is narrower confidence intervals without increasing computational time.

5.1.1.1 Implementations Issues

One practical issue arising from running stochastic simulations on supercomputers is statistical variation across parallel runs. Care must be taken to ensure each processor generates statistically independent runs. The simulations in this chapter use a different seed for each processor. Given that there are \( Q \) nodes in a simulation, MPI ranks each processor from 0 to \( Q - 1 \). Here, the seed is set to the processor rank.

Ref. [45] discusses an alternate way of generating statistically independent runs. Each time the simulations use a random number, \( Q \) values are generated (one for each processor). Because only one seed is used, there exists the possibility of exhausting the random number generator depending on how large the cycle value is. One way around this is resetting the seed periodically throughout the simulation. In addition, distributing values for each processor may mean longer runs from having to wait for each processor to generate a random number. Still, the extra computing time is probably negligible when compared to total run times.
<table>
<thead>
<tr>
<th>Step Size</th>
<th>$R$</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.0333</td>
<td>0.0487</td>
</tr>
<tr>
<td>0.0075</td>
<td>0.0320</td>
<td>0.0109</td>
</tr>
</tbody>
</table>

Table 5.1: Histogram distance between AMSA and SSA for $R$ and $P$ distributions.

Figure 5.1: Bar plots illustrating the relative amount of time spent in each possible DNA state.

Figure 5.2: Histogram of $R$ population for SSA (solid) vs. AMSA (dashed) with $\tau = 0.0075$ (left panel) and $\tau = 0.1$ (right panel).
Figure 5.3: Histogram of $P$ population for SSA (solid) vs. AMSA (dashed) with $\tau = 0.0075$ (left panel) and $\tau = 0.1$ (right panel).

Figure 5.4: Confidence intervals for histograms of $R$ distribution for SSA (solid) vs. AMSA (dotted) with $\tau = 0.0075$ (left panel) and $\tau = 0.1$ (right panel).
Figure 5.5: Confidence intervals for histograms of $P$ distribution for SSA (solid) vs. AMSA (dashed) with $\tau = 0.0075$ (left panel) and $\tau = 0.1$ (right panel).

5.1.2 Revisiting the lac System

Chapter 4 introduces a more detailed example of a genetic regulation system. The details can be found in tables 4.8 and 4.9. Chapter 4 uses runs from merely serial processors. To get a better idea of how product molecule populations vary, 100 runs of the exact stochastic algorithm are compared to that of AMSA (figure 5.6). These simulations are executed for only one generation. Again, simulations use 100 dual core 2.2 GHz AMD Opteron 275 CPUs.

These graphs demonstrate agreement between the two methods, a further indication of AMSA’s success. Like the previous section, a different seed is assigned to each processor.
Figure 5.6: Time course for 100 realizations of the \textit{lac} genetic regulation example (product molecules). SSA results are on the top, while those of AMSA are on the bottom. The graphs only represent one generation.
5.2 Lambda Phage Regulation: A Viral Infection Model

In bacterial processes similar to the self-regulation example, gene states are often Boolean in nature— they are either activated or inactivated. Some systems contain more complicated operator setups. However, in other examples, operator states actually compete. One example is lambda phage, a virus that preys on *E. coli*. Once inside the host, the virus can either destroy the host cell (lysis), or integrate its own DNA in that of the host’s (lysogeny) and wait for a while before initiating destruction.

Lysis is initiated by the cro gene. During transcription, messenger RNA coding for the molecule CRO is produced. CRO is capitalized to distinguish it from the corresponding gene. After translation, CRO may help activate other genes that code for the production virus-manufacturing proteins. Also, dimerized CRO (CRO₂) is able to latch back on to the operators and perpetuate CRO production. When CRO levels becomes too large, self-inhibition sets in and production decreases in a manner reminiscent of that of *P* in the self-regulation example.

Lysogeny, on the other hand, is initiated by the gene cl. The transcription process yields mRNA that is translated into CI molecules. Again, capitalization is used to avoid confusion. Once CI is dimerized (CI₂), it competes with CRO for attachment to operator sites. If successful, more CI molecules are created. CI also inhibits itself at large levels of expression. Illustrations outlining these mechanisms can be found in Ref. [43].
There are three operator sites: OR1, OR2, and OR3. CRO2 molecules attach themselves to OR3 first, OR2 second, and OR1 lastly. CI2 proceeds in the opposite fashion. When all operators are free, or a CRO dimer latches on to OR3, cro begins transcription resulting in more CRO molecules. If CRO dimers occupy both OR3 and OR2 or all three operator sites, CRO can’t be transcribed. For this model, CI is manufactured only when both OR1 and OR2 are occupied by CI2 molecules. Other operator states are possible, but are unlikely. Therefore, they aren’t included in this model.

One of CI and CRO eventually will overtake the other in terms of expression. If CRO dominates, lysis is initiated; but if CI dominates, lysogeny occurs. Since two possible outcomes are likely, yet another benchmark is provided with which to measure AMSA performance. Molecular population distribution is one way to compare hybrid models, but isn’t the only way. Comparisons of distributions across cellular populations is yet another way to compare approaches.

5.2.1 Modeling the Reaction System

The reactions in table 5.2 are based on the model in Ref ([43]). The associated initial conditions and parameters can be found in tables 5.3 and 5.4, respectively. CRO usually dominates, so 8 molecules of CI are given to push the odds in favor of lysogeny. [43] To obtain more realistic simulations, one should start both CI and CRO at 0.
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Rate Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CI \rightarrow \emptyset$</td>
<td>$c_1$</td>
<td>0.005s$^{-1}$</td>
</tr>
<tr>
<td>$2CI \rightarrow CI_2$</td>
<td>$c_2$</td>
<td>0.01(molec s)$^{-1}$</td>
</tr>
<tr>
<td>$CI_2 \rightarrow 2CI$</td>
<td>$c_3$</td>
<td>0.25s$^{-1}$</td>
</tr>
<tr>
<td>$D + CI_2 \rightarrow Di_1$</td>
<td>$c_4$</td>
<td>9.768(molec s)$^{-1}$</td>
</tr>
<tr>
<td>$Di_1 \rightarrow D + CI_2$</td>
<td>$c_5$</td>
<td>15s$^{-1}$</td>
</tr>
<tr>
<td>$Di_1 + CI_2 \rightarrow Di_2$</td>
<td>$c_6$</td>
<td>9.768(molec s)$^{-1}$</td>
</tr>
<tr>
<td>$Di_2 \rightarrow Di_1 + CI_2$</td>
<td>$c_7$</td>
<td>15s$^{-1}$</td>
</tr>
<tr>
<td>$Di_2 + CI_2 \rightarrow Di_3$</td>
<td>$c_8$</td>
<td>9.768(molec s)$^{-1}$</td>
</tr>
<tr>
<td>$Di_3 \rightarrow Di_2 + CI_2$</td>
<td>$c_9$</td>
<td>2022.38s$^{-1}$</td>
</tr>
<tr>
<td>$Di_2 + RNAP \rightarrow Di_2 + RNAP + CI$</td>
<td>$c_{10}$</td>
<td>0.5(molec s)$^{-1}$</td>
</tr>
<tr>
<td>$CRO \rightarrow \emptyset$</td>
<td>$c_{11}$</td>
<td>0.005s$^{-1}$</td>
</tr>
<tr>
<td>$2CRO \rightarrow CRO_2$</td>
<td>$c_{12}$</td>
<td>0.01(molec s)$^{-1}$</td>
</tr>
<tr>
<td>$CRO_2 \rightarrow 2CRO$</td>
<td>$c_{13}$</td>
<td>0.25s$^{-1}$</td>
</tr>
<tr>
<td>$D + CRO_2 \rightarrow Do_1$</td>
<td>$c_{14}$</td>
<td>9.768(molec s)$^{-1}$</td>
</tr>
<tr>
<td>$Do_1 \rightarrow D + CRO_2$</td>
<td>$c_{15}$</td>
<td>29.77s$^{-1}$</td>
</tr>
<tr>
<td>$Do_1 + CRO_2 \rightarrow Do_2$</td>
<td>$c_{16}$</td>
<td>9.768(molec s)$^{-1}$</td>
</tr>
<tr>
<td>$Do_2 \rightarrow Do_1 + CRO_2$</td>
<td>$c_{17}$</td>
<td>245.371s$^{-1}$</td>
</tr>
<tr>
<td>$Do_2 + CRO_2 \rightarrow Do_3$</td>
<td>$c_{18}$</td>
<td>9.768(molec s)$^{-1}$</td>
</tr>
<tr>
<td>$Do_3 \rightarrow Do_2 + CRO_2$</td>
<td>$c_{19}$</td>
<td>245.371s$^{-1}$</td>
</tr>
<tr>
<td>$D + RNAP \rightarrow D + RNAP + CRO$</td>
<td>$c_{20}$</td>
<td>0.5(molec s)$^{-1}$</td>
</tr>
<tr>
<td>$Do_1 + RNAP \rightarrow Do_1 + RNAP + CRO$</td>
<td>$c_{21}$</td>
<td>0.5(molec s)$^{-1}$</td>
</tr>
</tbody>
</table>

Table 5.2: Parameters for lambda phage model. For DNA states, $D$ indicates all three operator regions are unbound. $Di$ and $Do$ represent that operator regions are occupied by $CI$ and $CRO$, respectively. The subscript (e.g. the 2 in $Di_2$) indicates the total number of $CI$ or $CRO$ molecules are attached to the operator regions. The symbol $\emptyset$ denotes a degradation reaction.

Of course, this example doesn’t contain the entire range of reactions and molecular types involved in lambda phage infections. The system is far more complicated. Ref. [3] gives a more detailed model. Because the purpose of this section is introducing a model with two different outcomes, a simple model suffices. A similar application of stochastic simulations to the lambda phage model is also given in Ref. [15].
<table>
<thead>
<tr>
<th>Molecular Type</th>
<th>Initial Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI</td>
<td>8</td>
</tr>
<tr>
<td>CRO</td>
<td>0</td>
</tr>
<tr>
<td>RNAP</td>
<td>10</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>other species</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.3: Initial values for lambda phage model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Lambda Phage Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma_1$</td>
<td>0.1</td>
</tr>
<tr>
<td>$\Gamma_2$</td>
<td>177.77</td>
</tr>
<tr>
<td>$\Lambda$</td>
<td>10</td>
</tr>
<tr>
<td>$\rho$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 5.4: Parameters for lambda phage system.

5.2.2 Results

Figure 5.2.2 shows exact SSA runs of the lambda phage system for two different outcomes. In the first panel, lysogeny wins, as CI becomes the dominant molecule. In the second panel, the reverse is true, and lysis wins. As the graphs suggest, just because one pathway is initially dominant doesn’t imply it will remain so. The first graph shows CRO production ballooning at first, and remaining steady afterwards. In the second graph, CRO maintains dominance throughout the simulation. AMSA runs of the lambda phage example exhibit similar behaviors (figure 5.2.2). This time, in the lysogeny case has CI maintaining dominance throughout.

This analysis brings up a major point. Every bacterial cell is different, especially with respect to intracellular environment. Different molecular distributions can make one particular bacterium more subject to lysis than others. In the other genetic regulation examples, though protein levels differ, there still is only one
Figure 5.7: Molecular populations vs. time for instance in which CI dominates (left panel) and CRO dominates (right panel). The exact SSA is used for these simulations.

Figure 5.8: Molecular populations vs. time for instance in which CI dominates (left panel) and CRO dominates (right panel). AMSA is the used for these simulations.
Figure 5.9: Histogram of CI dimer populations after time $T=200s$ for the SSA (solid) and AMSA (dashed) with $\tau = 0.1$.

major type of product molecule. Statistical analysis is limited to the amounts of time spent in each state. On the other hand, the lambda phage system exhibits statistical variation across cells as well as inside the cell.

Therefore, it's logical to consider each supercomputer processor as representing a different bacterium. If AMSA provides a decent approximation to Gillespie's algorithm, the frequency with which the exact SSA chooses lysogeny should be the same as that of AMSA, given a large enough sample size. Figures 5.9 and 5.10 show the CI distributions at $T = 200s$ for two different choices of time step. For $\tau = 0.1$, the AMSA distribution matches up quite well with that of the SSA at low levels. At higher levels, the graph is shifted to the left. At a smaller time step, the smoothed histograms match up quite well, as expected. The graphs in figures 5.11 and 5.12 show CRO distributions for $\tau = 0.1$ and $\tau = 10^{-4}$. Results are similar to
Figure 5.10: Histogram of CI dimer populations after time T=200s for the SSA (solid) and AMSA (dashed) with $\tau = 10^{-4}$.

Figure 5.11: Histogram of CRO dimer populations after time T=200s for the SSA (solid) and AMSA (dashed) with $\tau = 0.1$. 
Figure 5.12: Histogram of CRO dimer populations after time $T=200s$ for the SSA (solid) and AMSA (dashed) with $\tau = 10^{-4}$.

those of CI.

Figure 5.13 shows the $L_1$ error vs. time step for AMSA to give a better idea of how much accuracy depends on the choice of time step. This quantity is calculated by summing the distances between the distributions for AMSA and the SSA. Notice the sudden jump after the first value. One hypothesis of the sharp increase in accuracy at $\tau = 10^{-4}$ is sudden changes in propensity values. Upon examination of the rate constants in table 5.2, one finds that the propensities of reactions involving DNA states could go from 0 to over 2000 quickly. For example, that there were noticeably fewer steps in which the algorithm had to back up because a trajectory fell below 0. These simulations utilized the RTC, a cluster with Intel Itanium 2 900 MHz processors. There were 1000 runs for each time step.

The issue is further illustrated in table 5.5. The amount of time it takes to
<table>
<thead>
<tr>
<th>Method</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSA</td>
<td>4.57 min</td>
</tr>
<tr>
<td>AMSA (τ = 0.0001)</td>
<td>10.51 hrs</td>
</tr>
</tbody>
</table>

Table 5.5: Analysis of run times for 1000 runs of viral infection system. Simulations were run serially.

![](image)

Figure 5.13: (Left Panel) $L_1$ norm accuracy vs. time step for CI dimer. (Right Panel) $L_1$ norm accuracy vs. time step for CRO dimer.

generate 1000 runs in serial for $\tau = 10^{-4}$ dwarfs that for the corresponding SSA run.

A dilemma has in fact arisen since $10^{-4}$ is close to the largest time step for which AMSA runs are reasonably accurate.

To test the hypothesis, some of the DNA states were "removed". Deleting $D_0_1$, $D_0_2$, $D_0_3$, and $D_1_3$ from the system not only exacerbates the dominance of the system by the gene states, but also avoids dramatic shifts in propensity values. Even though most of the states involving CRO were removed, there still is the possibility of lysis. The Cro gene can transcribe CRO without the attachment CRO dimer. Deletion was executed by setting the rate constants of all reactions creating those states to 0. To push the system more in favor of lysis, the initial state of CI
Table 5.6: Parameters for modified lambda phage reaction system.

was lowered to 5. The list of reactions with modified constants can be found in table 5.6. The corresponding initial conditions are in table 5.7.

The time step \( \tau = 0.025 \) seemed to give good accuracy and timing results. Accuracy comparisons can be found in figure 5.14. Timing comparisons in table 5.8 indicate that here, AMSA performs much better than without the state deletions. It can be concluded that though AMSA captures reaction system dynamics well for the previous two genetic regulation systems, it falters when applied to examples dominated by reactions involving small amounts of molecules and large propensities.
<table>
<thead>
<tr>
<th>Molecular Type</th>
<th>Initial Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI</td>
<td>5</td>
</tr>
<tr>
<td>CRO</td>
<td>0</td>
</tr>
<tr>
<td>RNAP</td>
<td>10</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>other species</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.7: Initial values for modified lambda phage model.

<table>
<thead>
<tr>
<th>Method</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSA</td>
<td>0.69 min</td>
</tr>
<tr>
<td>AMSA ($\tau = 0.025$)</td>
<td>3.43 min</td>
</tr>
</tbody>
</table>

Table 5.8: Analysis of run times for 1000 runs of modified viral infection system. Simulations were run serially.

Figure 5.14: (Left Panel) Histogram of CI dimer population for modified system. (Right Panel) Histogram for CRO dimer population.
Chapter 6

Conclusion

The study of simulation of molecular populations in chemical reactions systems has blossomed the last couple of decades. In particular, Gillespie’s exact stochastic simulation algorithm and its perceived importance in biological applications stemmed new interest in the field. [13] Though effective, it’s slow computational performance was a concern, especially since cellular reaction systems are often stiff and contain channels firing at widely varying time rates.

Many articles address hybrid algorithms as a remedy to the slow computational speed of Gillespie’s exact stochastic simulation algorithm. Haseltine and Rawlings introduce a method that uses differential equations for faster reactions and the direct method for slower ones. Burrage, Tian, and Burrage include the direct method, $\tau$-leaping, and the chemical Langevin equation in their method. These are only two of many versions of reaction system modeling.
Despite the recent developments, there's still room for improvement in multi-scale simulation. The AMSA approach offers such an improvement. A modified version of the first reaction method for slow reactions is combined with differential equations, stochastic and deterministic. Fast reactions involving species small in number are modeled with a Poisson random variable. Suggestions were proposed regarding time step, threshold choices, and avoiding negative populations. Applications to a genetic self-regulation toy model and a more realistic example involving the lac system demonstrate that AMSA provides much faster simulations than those of the SSA with acceptable accuracy.

One key issue in the field of multi-scale simulations is the trade-off between computational efficiency and accuracy. In AMSA, the most important parameters that tune accuracy are time step and the lower reaction propensity threshold. Dynamic partitioning gives another angle from which to attack the problem. [21] Often, the success of a particular hybrid model depends highly on the choice of time step and thresholds.

Discretion must be exercised when comparing multi-scale timing results with that of the SSA. The true measure of speedup can only be determined when the exact method is at its most efficient. Here, an efficient version of the direct SSA is used as the gold standard. A significant contribution to the field would involve implementing the various hybrid models on a benchmark example to determine which ones work best.
The seemingly exponential increase in the amount of literature discussing ways to speed up stochastic simulation of chemical reaction systems is a testament to its importance in biology as well as other fields. Our efficient algorithm’s blend of considerable speedup and accuracy hopefully is a step in the right direction.

6.1 On the Use of Multi-processor Machines

Supercomputers are multi-processor machines that allow several hundred runs of stochastic and multi-scale simulations simultaneously. They allow for larger amounts of data to be generated in smaller times. They also provide a better benchmark with which to measure accuracy by reducing statistical variation with larger sample sizes.

6.2 Future Directions

Then, the question of biological accuracy arises. Does the data from multi-scaled simulations match up with real data from in vivo systems? Can the hybrid models then be used to make conjectures resulting in new biological results? Should multi-scale methods such as AMSA be accurate enough, they should provide as much information about reaction systems as the exact SSA.

There have even been ventures into using algorithms such as the SSA to model spatio-temporal aspects of molecular simulations. [8] Monte Carlo and molecular
dynamics simulations can be inefficient and pose CPU time issues that rival that of even the most naive implementations of the SSA. Perhaps multi-scale models can assist in these areas as well.

In conclusion, there is a lot left to be discovered. Hopefully, future research will at least put a dent in this inscience.
Appendix

AMSA Code

C***********************************************************************
C
C Program amsa.f: Adaptive Multi-Scale Simulation Algorithm
C
C This is AMSA. It accepts an input file that contains initial conditions, parameters, and stoichiometric matrices and outputs a file containing a matrix of trajectories and a file containing histogram information.
C
C Input Files - buf5
C Output Files - amsa1.out
C amsalb.out
C***********************************************************************

program amsa
include 'mpif.h'

character*100 vars

integer ierror, rank, size, st, ix

integer i, j, k, d, idx, mu, iseed, th, LP(100, 100), L0(100, 100)
integer Z(8, 100), zbqlpoi, ignpoi, rn
real*8 XL(100, 100), LR(100, 100), L(100, 100), R(100, 100)
real*8 t, Tt, ran2, snorm, a0, sig, X0(100), X(100), t0, ti, tz, cc, Y(100)
real*8 r1,r2,dt,a(100),h(100),c(100),adt(100),ah(2),zbqlu01
real*8 ndx(100),tdx(100),sdx(100),odx(100),ot,nt,zbqlnor,Y0(100)
real*8 nto(100),ntg(100),B(1200,4),F(1000000,2),tmp,XX(10**7,5)

character(LEN=32) :: fn, fm, fp
character(LEN=4) :: indx2

rn=0  ! bumps rank up by 'rn'

call MPI_INIT(ierr)                ! initialize MPI

call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call ZBQLINI(rank+rn)

call cpu_time_(t0)                 ! initialize CPU time
call cpu_time(ti)
t=0  ! initialize time
a0=0  ! total reaction "probability"
k=0

d=0

write indx2,'(I4)') rank+rn        ! make filename a character
ix=1
do i=1,4
   if (indx2(i:i) .ne. ' ') then
      ix=i
      exit
   endif
endo
if (rank+rn .lt. 5) then
   fn='amsa'//indx2(ix:4)//'.out'
endif
fm='amsa'//indx2(ix:4)//'b.out'
if (rank+rn .lt. 5) then
   open (unit=1, file=fn)
endif
open(2,file='buf5')
open (unit=3, file=fm)

do i=1,4
   do j=1,12000
B(j,i)=0.

Enddo
Enddo

Dt=1D0 ! time step
!Dt=5D0 ! alternate time step
Ot=dt
Th=10 ! abundance threshold
Ah(1)=1D0 ! lower propensity threshold
Ah(2)=177.7777D0 ! upper propensity threshold

Read(2,*)(M,N,Tf) ! read in data from 'buf5'
Read(2,*)(C(i),i=1,M)
Do i=1,M+1
   Read(2,*)(XL(i,j),j=1,N)
Enddo
Do i=1,M
   Read(2,*)(R(i,j),j=1,N)
Enddo

Do i=1,M
   Do j=1,N
      X(j)=XL(1,j)
      L(i,j)=XL(i+1,j)
      LR(i,j)=-L(i,j)+R(i,j);
      If (abs(LR(i,j)).gt.0) Then
         Lp(i,j)=1
      Else
         L0(i,j)=Th+1
      Endif
      X@0(j)=X(j)
   Enddo
Enddo
Call comb(h,X,L,M,N)
Do j=1,M
   Adt(j)=h(j)*c(j)*dt
   A0=A0+a(j)
   Ndx(j)=1D0
Enddo

Call clas(X,th,adt,dt,ah,LP,L0,Z,M,N) ! classify reactions
do i=1,Z(1,100)
   ntg(Z(1,i))=log(zbqlu01(0.0D0))
endo

do i=1,Z(2,100)
   ntg(Z(2,i))=log(zbqlu01(0.0D0))
endo

do while (t .lt. Tf) ! complete all three 'zones'
do i=1,Z(1,100) ! perform SSA
   nto(Z(1,i))=ntg(Z(1,i))
   ntg(Z(1,i))=ntg(Z(1,i))+adt(Z(1,i))
   if (ntg(Z(1,i)).gt.0) then
      ndx(1)=1D0
      call xta(ndx(1),X,LR,Z(1,i),N)
      ntg(Z(1,i))=log(zbqlu01(0.0D0))
   elseif (ntg(Z(1,i)).eq.0) then
      ntg(Z(1,i))=log(zbqlu01(0.0D0))
   endif
endo

do i=1,Z(2,100)
   nto(Z(2,i))=ntg(Z(2,i))
   ntg(Z(2,i))=ntg(Z(2,i))+adt(Z(2,i))
   if (ntg(Z(2,i)).gt.0) then
      ndx(2)=1D0
      call xta(ndx(2),X,LR,Z(2,i),N)
      ntg(Z(2,i))=log(zbqlu01(0.0D0))
   elseif (ntg(Z(2,i)).eq.0) then
      ntg(Z(2,i))=log(zbqlu01(0.0D0))
   endif
endo

do i=1,Z(3,100) ! perform Poisson r.v. step
   tdx(Z(3,i))=zbqlpoi(adt(Z(3,i)))
   call xta(tdx(Z(3,i)),X,LR,Z(3,i),N)
endo

do i=1,Z(4,100)
   tdx(Z(4,i))=zbqlpoi(adt(Z(4,i)))
   call xta(tdx(Z(4,i)),X,LR,Z(4,i),N)
endo

do i=1,Z(6,100) ! perform Euler-Maruyama
   sdx(Z(6,i))=adt(Z(6,i))+sqrt(adt(Z(6,i)))*zbqlnor(0.0D0,1.0D0)
   call xta(sdx(Z(6,i)),X,LR,Z(6,i),N)
endo
do i=1,Z(8,100)  ! perform Euler
   odx(Z(8,i)) = adt(Z(8,i))
   call xta(odx(Z(8,i)),X,LR,Z(8,i),N)
enddo

   do j=1,N        ! check to see if any X(j)<0
      if (X(j).lt.th) then
         if (X(j).lt.0) then
            nt=min(dt,.5D0*X0(j)*dt/(X0(j)-X(j)))
            do i=1,j
               X(i)=X0(i)! reset all populations to prev. values
            enddo
            do i=j+1,N  ! make sure to find 1st traj that hit 0
               if (X(i).lt.0) nt=min(nt,.5D0*X(i)*dt/(X(i)-X0(i)))
               X(i)=X0(i)
            enddo
            dt=nt
         endif
      endif
   enddo
   do i=1,M
      adt(i)=h(i)*c(i)*dt ! calculate propensities
   enddo
   call clas(X,th,adt,dt,ah,LP,LO,Z,M,N)
   do i=1,Z(1,100)
      ntg(Z(1,i))=nto(Z(1,i))
   enddo
   do i=1,Z(2,100)
      ntg(Z(2,i))=nto(Z(2,i))
   enddo
   goto 10
endif
enddo

   do j=1,N
      if (t.gt.10000) then
         B(int(X0(j))+1,j)=B(int(X0(j))+1,j)+dt
      endif
      if ((j.eq.1).and.(t.gt.100)) then
         if ((X(1).eq.X0(1))) then
            tmp=tmp+dt
         else
            k=k+1
            F(k,2)=X0(1)
            F(k,1)=tmp
         endif
      endif
   enddo
tmp=∅
endif
endif
X0(j)=X(j)  ! reset all populations to prev. values
enddo

call comb(h,X,L,M,N)

t=t+dt
dt=ot  !important

do i=1,M
   adt(i)=h(i)*c(i)*dt  !calculate propensities
endo

call clas(X,th,adt,dt,ah,LP,L0,Z,M,N)

if ((rank+rn.lt.5).and.(t.gt.1000)) then !calc. histograms
   write(1,'(F16.10,4F8.0)') t-1000,X(1),X(2),X(3),X(4)
endif
d=d+1
do j=1,4
   XX(d,1)=t
   XX(d,j+1)=X(j)
endo
do j=1,N
   if (X(j).lt.th) then
      if ((X(j)-int(X(j))).lt.zbqlu01(0.0D0)) then
         X(j)=1.*int(X(j))
      else
         X(j)=1.*int(X(j))
      endif
   endif
X0(j)=X(j)
endo
dooo

do k=2,d
   do j=1,4
      Y0(j)=XX(k-1,1)
Y0(j+1) = XX(k-1,j+1)
Y(1) = XX(k,1)
Y(j+1) = XX(k,j+1)
if (Y(1).gt.1000) then
    B(int(Y0(j+1))+1,j) = B(int(Y0(j+1))+1,j)+(Y(1)-Y0(1))
endif
enddo
dndo
doi = 1,1200
    write(3,*) (B(i,j)/(Tf-1000),j=1,4)
dendo
doi = 1,k
    write(4,*) F(i,1),F(i,2)
dendo
call MPI_FINALIZE(ierror)
close(1)
end

subroutine clas(X,th,a,dt,ah,LP,L0,Z,M,N) ! classify reacs.
    integer th,i,j,k,LP(100,100),L0(100,100),Z(8,100),W(100)
    real*8 a(100),ah(2),X(100),mn,dt,ran2
    do i = 1,8
        do j = 1,100
            Z(i,j) = 0 ! Z(i,100) = 0
        enddo
    enddo
doi = 1,M
    mn = LP(i,1)*X(1)+L0(i,1)
do j = 2,N
        mn = min(mn,LP(i,j)*X(j)+L0(i,j))
    enddo
    if (mn.lt.th) then ! microscale species
        if (a(i).gt.ah(2)) then ! fast reactions
            W(i) = 4
        elseif (a(i).gt.ah(1)) then ! medium reactions
            W(i) = 3
        else
            W(i) = 1 ! slow reactions
        endif
    elseif (a(i).gt.ah(2)) then ! macroscale species
        W(i) = 8 ! fast reactions
elseif (a(i).gt.ah(1)) then
    W(i)=6 ! medium reactions
else
    W(i)=2 ! slow reactions
endif
endif

Z(W(i),100)=Z(W(i),100)+1
Z(W(i),Z(W(i),100))=i
enddo
end

subroutine xta(fdt,X,LR,id,N) ! matrix/vector multiply
integer j,k,id,mn,N,Z(8,100)
real*8 X(100),LR(100,100),fdt
do j=1,N
    X(j)=X(j)+fdt*LR(id,j)
enddo
end

subroutine comb(h,X,L,M,N) ! calculate propensities
integer mu,j,k
real*8 h(100),X(100),L(100,100)
do mu=1,M
    h(mu)=1
    do k=1,N
        j=1
        do while (j .le. L(mu,k))
            h(mu)=h(mu)*((X(k)-(j-1))/j)
            j=j+1
        enddo
    enddo
enddo
enddo
end
SSA Code

C****************************************************************************************************************************C
C C Program stoch.f: Stochastic simulation algorithm C
C C This is Gillespie's exact stochastic simulation algorithm. C
C C It accepts an input file that contains initial conditions, C
C parameters, and stoichiometric matrices and outputs a file C
C containing a matrix of trajectories and a file containing C
C histogram information. C
C C Input Files - buf5 (reads in parameters) C
C Output Files - stoch1.out (holds time trajectories) C
C stoch1b.out (holds histograms) C
C C****************************************************************************************************************************C

program stoch
include 'mpif.h'
character*100 vars

integer ierror, rank, size, st, ix, f(20,20), d
integer i, M, N, j, k, mu, iseed, L(20,20), R(20,20), XL(20,20), LR(20,20)

double precision t, Tf, ran2, a0, r1, r2, tao, sig, X(20), X0(20)
double precision a(20), h(20), c(20), B(1200, 4), Fs(1000000, 2), tmp
double precision Y(1e6, 20), zbqlu01

character(LEN=32) :: fn, fm, fp
character(LEN=4) :: indx2

call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierror)
call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierror)
call ZBQLINI(rank) ! get processor rank

t=0  ! initialize time
tao=0  ! time step
a0=0 ! total reaction "probability"
k=0
d=0
tmp=0

write(indx2,'(I4)') rank ! make filenames a character
ix=1
do i=1,4
   if (indx2(i:i) .ne. ' ') then
      ix=i
      exit
   endif
endo
if (rank.lt.5) then
   fn='stoch'//indx2(ix:4)//'.'out'
endif
fm='stoch'//indx2(ix:4)//'b.out'
do i=1,4
   do j=1,1200
      B(j,i)=0. ! histogram matrix
   enddo
endo
open(10,file='buf5')

read(10,*) M,N,Tf ! read in # of reactions, # of species, ! and final time
read(10,*) (c(i),i=1,M) ! set of rate constants
do i=1,M+1
   read (10,*) (XL(i,j),j=1,N) ! initial state and ! reactant stoichiometric matrix
endo
do i=1,M
   read (10,*) (R(i,j),j=1,N)! product stoichiometric matrix
endo

do i=1,M
   do j=1,N
      X(j)=XL(1,j)
      X0(j)=X(j)
      L(i,j)=XL(i+1,j)
   enddo
enddo
LR(i,j)=-L(i,j)+R(i,j) ! update stoichiometric matrix
enddo
endo
call affect(f,L,LR,M,N) ! determines which reactions affect
! which propensities
endo
do i=1,M
   call comb(h,f,i,X,L,N) ! find total # ways a reaction can
   ! occur
endo
do j=1,M
   a(j)=h(j)*c(j) ! calculate reaction sum
   a0=a0+a(j)
endo
if (rank.lt.5) then ! open input and output files
   open (unit=1, file=fn)
endif
open (unit=3, file=fm)
endo
do while (t .lt. Tf)
   r1=zbqlu01(0.0D0) ! random numbers from unif(0,1)
   r2=zbqlu01(0.0D0)
   tao=log(1/r1)/a0 ! simulate next reaction time
   mu=0 ! reset reaction index
   sig=0 ! reset propensity sum
   do while (sig .lt. r2*a0) ! simulate which reaction occurs
      mu=mu+1
      sig=sig+a(mu)
   enddo
   t=t+tao ! update time
endo
do j=1,N
   X(j)=X(j)+LR(mu,j) ! adjust molecular populations
endo
do j=1,N ! update histograms
   if (t.gt.1000) then ! update time bin for each species
      B(int(X0(j))+1,j)=B(int(X0(j))+1,j)+tao
   endif
endo
X0(j)=X(j) ! reset old state
if ((rank.lt.5).and.(t.gt.1000)) then
    if ((t.lt.2000).or.(t.gt.20000)) then
        write(1,'(F16.10,4F8.0,I4)') t-1000, (X(i),i=1,N), mu
    endif
    ! write state info to file (line above)
endif

call comb(h,f,mu,X,L,N)  ! calculate combinatorial entities

do i=1,f(mu,20)  ! only update propensities affected by
    a(f(mu,i))=h(f(mu,i))*c(f(mu,i))  ! occurring reaction
endo

a0=0;
do i=1,M  ! calculate propensity sum
    a0=a0+a(i)
endo

do i=1,12000  ! write histogram information to file
    write(3,*)(B(i,j)/(Tf-1000),j=1,4)
endo

call MPI_FINALIZE(ierr)
close(1)
end

subroutine comb(h,f,id,X,L,N)  ! calculate # of ways reaction
    integer i,j,k,mu,id,f(20,20),L(20,20),N  ! could occur
double precision h(20),X(20)
do mu=1,f(id,20)
h(f(id,mu))=1
do k=1,N
    j=1
    do while (j .le. L(f(id,mu),k))
        h(f(id,mu))=h(f(id,mu))*((X(k)-(j-1))/j)
        j=j+1
    enddo
endo
endo
end
subroutine affect(f,L,LR,M,N)
! find which reactions affect which
integer f(20,20), g(20,20), L(20,20), LR(20,20), M, N, i, j, k
do j=1,20
  do k=1,20
    g(j,k)=0 ! set up adjacency matrix
  enddo
enddo
do j=1,M
  do k=1,N
    if (abs(LR(j,k)).gt.0) then
      do i=1,M
        if (L(i,k).gt.0) g(j,i)=1 ! find adjacency matrix
      enddo
    endif
  enddo
enddo
do j=1,M
  f(j,20)=0
  do i=1,M
    if (g(j,i).eq.1) then
      f(j,20)=f(j,20)+1 ! matrix of indices of adj matrix
      f(j,f(j,20))=i
    endif
  enddo
enddo
end

C**********************************************************************C
C
C Program buf5: Input file for AMSA and SSA
C
C Note: Comments are for illustration purposes only, and should be deleted before running code.
C
C**********************************************************************C

6 4 21000. ! # of reac., species, final time
0.0001 0.075 20 50 5 0.5 ! reaction constants
<table>
<thead>
<tr>
<th>1</th>
<th>406</th>
<th>1</th>
<th>40</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>! initial conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>! reactant stoichiometric matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>! product stoichiometric matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Bibliography


