MODELING, SIMULATION, AND VERIFICATION OF BIOCHEMICAL PROCESSES USING STOCHASTIC HYBRID SYSTEMS

By

Derek Riley

Dissertation
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
of the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

May, 2009

Nashville, Tennessee

Approved:
Professor Xenofon Koutsoukos
Professor Janos Sztipanovits
Professor Gabor Karsai
Professor Gautam Biswas
Professor Larry Dowdy

ACKNOWLEDGMENTS

I am grateful to all of those people with whom I have had the pleasure of working during my time at Vanderbilt. I would like to especially thank my adviser Xenofon Koutsoukos for his guidance, help, and patience with this work. I am also grateful to the members of my committee: Janos Sztipanovits, Larry Dowdy, Gabor Karsai, and Gautam Biswas for their insight and help. I would also like to thank Heath LeBlanc for helping me edit and prepare this dissertation.

I would also like to thank my family for their support of my research, and I would like to specifically thank my sister for her collaboration and input in this work. Also, this work would not have been possible without the financial support of the National Science Foundation CAREER CNS-0347440 grant awarded to Xenofon Koutsoukos.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Formal modeling and analysis methods hold great promise to help further discovery and innovation for biochemical systems. Domain experts from physicians to chemical engineers can use computational modeling and analysis tools to clarify and demystify complex systems. Models can be tested and adapted inexpensively providing new insights. However, development of accurate and efficient modeling methodologies and analysis techniques are open challenges for biochemical systems.

Biochemical systems often contain continuous, discrete, and stochastic dynamics, so accurate models must capture these dynamics as well as potential interactions between them [70, 98]. This work is based on the Stochastic Hybrid Systems (SHS) modeling paradigm. SHS provide a formal framework that combines Stochastic Differential Equations (SDEs) with discrete dynamics to capture stochastic continuous dynamics, deterministic transitions, and probabilistic transitions. Further, simulation and other analysis methods for SHS are promising for the study of complex systems [61, 81].

In this dissertation we develop a SHS framework for modeling and analysis of biochemical systems. We use realistic case studies to demonstrate the modeling capabilities of SHS and our proposed methodologies. These case studies include models of sugar cataract development in the lens of a human eye [120], a commercial biodiesel production system [126], glycolysis, which is a cellular energy conversion mechanism found in every living cell [125], and the water and electrolyte balance system in humans [123].

We develop SHS analysis methods that include algorithms for simulation, verification, and Monte Carlo-based reachability analysis. We develop accurate, efficient simulation methods for SHS using both fixed step and adaptive step algorithms. We also present an exhaustive verification method based on dynamic programming that can be used to determine reachability or safety probabilities for every state of the system [83], as well as a parallelization method that allows the application of the approach to realistic systems [122]. For systems too large for exhaustive verification we develop Monte Carlo-based

reachability analysis methods [123]. We extend Monte Carlo methods with a rare event variance reduction method called MultiLevel Splitting (MLS) to improve accuracy and efficiency [125, 126].

Another contribution of this work is the software tools developed. The analysis methods are implemented and tested using our case studies to demonstrate performance and efficiency with realistic systems. Software implementations of the modeling methodology and the analysis methods provide a platform for testing existing models as well as new models. Parallelized versions of both the exhaustive verification method as well as the Monte Carlo methods are developed to improve efficiency and enhance the applicability of the methods to large systems.

In the rest of this chapter, we describe the motivation for our work followed by challenges and specific research objectives. We then discuss the contributions of this dissertation, and conclude with the organization of the rest of the dissertation.

Motivation

As biomedical research advances into more complicated systems, there is an increasing need to model and analyze these systems to better understand them. Realistic biochemical systems often involve continuous, discrete, and stochastic dynamics [128], so it is important to use a modeling paradigm that can capture all of these dynamics in an extensible, formal framework.

The modeling and analysis tools that are currently available are typically designed for specific biochemical systems and are limited in accuracy or scalability [11, 18, 101]. Further, many current analysis techniques are not built upon formal methods [128, 70], and efficiency challenges often limit the application of the techniques to realistic systems. Therefore, it is important to create formal, accurate, and scalable modeling techniques in addition to accurate and efficient analysis tools to further the understanding of complex biochemical systems.

Simulation is a powerful analysis tool because it can expose the behavior of a system in conditions where traditional experimentation is costly or impossible. Accurate and efficient simulation of biochemical systems based on SHS is difficult because of the inherent error that

is introduced by the interplay of continuous, discrete, and stochastic dynamics. Accuracy can usually be improved by using finer approximation methods, but this comes at the cost of efficiency. Adaptive time stepping is an effective method for improving the efficiency of simulation methods, but it poses difficulty for SHS because of the challenges of error estimation methods for stochastic dynamics [90].

Simulation methods are useful for establishing potential outcomes, but often a more comprehensive understanding of the system is required. Exhaustive verification methods can be used to formally compute properties such as reachability or safety for every state of the system [83, 9, 20]. Verification of reachability or safety properties for biochemical systems is a critical problem because the results of the verification can help scientists gain insights into the modeled systems, expose flaws, or validate the intended function more robustly than simple simulations [15, 14, 59].

While exhaustive verification techniques are useful for small systems, usually they are subject to the curse of dimensionality and are not scalable for large systems [120]. An alternative to exhaustive verification algorithms is Monte Carlo methods, which can be used to determine reachability or safety properties for stochastic systems. Monte Carlo methods utilize multiple individual simulations from a single initial state to determine reachability probabilities for that state [125].

Efficiency and accuracy of Monte Carlo methods are degraded significantly when rare events are present, so variance reduction methods are necessary to maintain accuracy. MLS is an adaptation of Monte Carlo methods that can be used to reduce the variance and increase the efficiency of the estimator when a rare event is present [53]. While MLS holds great promise to improve accuracy and efficiency of Monte Carlo methods, it also poses challenges in determining appropriate parameters for optimal performance. Monte Carlo and MLS methods can be parallelized to further improve efficiency.

Challenges

There are many open problems in the fields of modeling and analysis of biochemical systems. We focus on the challenges that relate to computational aspects of modeling and analysis methods.

Modeling realistic systems Many simple, small biochemical models exist, but larger, more accurate, realistic models are rare because they are significantly more difficult to develop and analyze. Modeling methodologies typically require dynamics and parameters to be fully specified to create a valid model. Once a model has been created, the model should be thoroughly tested and compared to experimental data to validate the correctness of the model. Validation can be challenging because error can be introduced in the modeling method, the provided parameters, or the analysis technique. Further, experimental results are not available for many biochemical systems because of physical or ethical restrictions, so experimental-based model validation is not possible for every model.

Simulation methods Simulation is challenging for SHS because of the interplay of the discrete, continuous, and stochastic dynamics. Discrete transitions can cause discontinuities that can create significant error if handled improperly. Detecting boundary crossing in the presence of stochastic dynamics is also inherently inaccurate. Stochastic dynamics must also be carefully simulated to ensure bias is not introduced. Existing fixed time stepping methods require very small time steps to generate accurate approximations, which is computationally expensive. Adaptive time stepping methods have been developed for SDEs, but error estimation methods can decrease accuracy significantly if not handled carefully.

Reachability analysis of SHS Exhaustive verification methods can be used to determine reachability properties for SHS, but they are either designed for special cases, or they are computationally expensive. General exhaustive verification techniques are plagued by the curse of dimensionality, which dictates that as the state space grows linearly, the computation required to analyze the system grows exponentially. To date, only simple examples have been used to demonstrate verification methods.

Monte Carlo analysis can also be used to determine reachability properties for SHS, but when Monte Carlo methods are used with inaccurate simulation trajectories, the accuracy of the results are compromised. Pseudo-random number generators can also introduce bias into Monte Carlo estimates that must be handled. Further, Monte Carlo methods can become prohibitively inefficient and inaccurate for systems with rare events, so variance reduction methods may be necessary. Variance reduction methods are challenging for SHS because

of discontinuities caused by discrete transitions. Also, variance reduction methods are controlled by adjustable parameters that require tuning to achieve good results. Methods for determining the optimal parameter settings do not exist for high-dimensional systems. Even with variance reduction methods, some rare events may not be accurately estimated.

Interdisciplinary bridge One of the significant challenges for modeling realistic systems is the interdisciplinary nature of the task. Systems that would benefit from modeling typically require extensive domain knowledge, and methods that can provide the modeling and analysis tools require extensive knowledge to implement and use.

Contributions

This dissertation presents realistic biochemical case studies along with accurate and efficient computational analysis methods and experimental results.

1. **Modeling realistic biochemical systems** This dissertation presents a modeling methodology for biochemical systems using SHS. Biochemical processes are inherently stochastic and often contain both continuous and discrete behavior, so SHS are an ideal modeling paradigm for capturing their complex dynamics. We present several realistic SHS models of various sizes and complexity.

   We present models of (a) sugar cataract development in the lens of a human eye, (b) a commercial biodiesel production system, (c) glycolysis, which is a cellular energy conversion mechanism found in every living cell, and (d) the water and electrolyte balance system in humans. We present these models as case studies to demonstrate the effectiveness and accuracy of our modeling methodology. We validate the sugar cataract development model using a highly accurate fine-grained simulation method called the stochastic simulation algorithm. We also validate the biodiesel model using published results collected from an experimental processor.

2. **Simulation methods for SHS** We develop an advanced simulation technique for SHS that employs improved boundary crossing detection methods for absorbing and reflecting boundaries using probabilistic sampling. Further, we develop and present an adaptive time stepping simulation method for SHS that improves the accuracy and

efficiency. Experimental results of the improved simulation methods using our case studies are also presented to demonstrate the performance and accuracy improvements of these methods.

3. **Verification of SHS** We present an exhaustive verification method based on dynamic programming that can be used to determine reachability or safety probabilities for SHS. The verification method suffers from the curse of dimensionality, so we develop a parallel dynamic programming implementation of the verification algorithm that improves the performance of the algorithm. Although scalability is a limiting factor, this work demonstrates that the parallel technique is feasible for realistic biochemical systems.

4. **Reachability analysis using Monte Carlo and variance reduction methods** We develop Monte Carlo methods for SHS that utilize our advanced simulation methods as well as parallel computing techniques to further enhance accuracy and efficiency. We develop a MLS variance reduction method for SHS that improves the accuracy and efficiency of Monte Carlo methods in the presence of rare events. Results for the biodiesel and glycolysis models using MLS are presented to demonstrate its effectiveness for realistic systems. We also present MLS parameter selection methods and experimental results to help choose appropriate MLS configuration parameters.

Organization

Chapter *II* presents the related work to provide a survey of the current research. Stochastic hybrid modeling paradigms, methods for modeling biochemical systems, and non-stochastic hybrid modeling paradigms are presented. Several methods for simulating stochastic dynamical systems as well as verification methods are also discussed. Monte Carlo approaches including variance reduction methods are presented to provide context for our later work. The chapter is completed with a comparison of the related work and the contributions of the dissertation.

Chapter *III* presents our modeling methodology and the biochemical models we developed. The modeling methodology is designed to model any biochemical system with known

chemical reactions and kinetic coefficients. The method is demonstrated for several models including sugar cataract development in the eyes of humans, a biodiesel production system, glycolysis, a cellular energy conversion process, and water electrolyte balance in humans.

Chapter IV presents simulation methods for SDEs and probabilistic transitions as well as detection methods for absorbing boundaries and reflecting boundaries. Improved stochastic methods for absorbing and reflecting boundaries are presented as well as high-order methods for simulating SDEs. These methods are combined to create improved simulation algorithms for SHS. An error estimation method is developed as well as an adaptive SHS simulation algorithm. Simulation results are presented from the sugar cataract development model, biodiesel model, and water balance model to demonstrate the performance and accuracy improvements of our simulation methods.

Chapter V presents our exhaustive verification method for SHS based on dynamic programming. It is computationally expensive for high-dimensional systems, so we develop a parallel implementation of the verification algorithm, and we present experimental results for a navigation benchmark and room heater benchmark. We also present results for the sugar cataract development system and biodiesel model.

Chapter VI presents Monte Carlo methods for SHS. The reachability problem for SHS is formulated, and it is shown how Monte Carlo methods can be used to solve the reachability problem. The MLS rare event detection method for SHS is developed along with a discussion of parameter configuration for improved variance reduction. The chapter continues with experimental results demonstrating methods for choosing MLS parameters and a comparison of the MLS results with the results generated using our dynamic programming exhaustive verification methods.

Chapter VII summarizes the contributions of this dissertation. Limitations of the approach are described, and future directions are also discussed.

CHAPTER II

RELATED WORK

In this chapter we present research related to modeling and analysis of Stochastic Hybrid Systems (SHS) and biochemical systems. We compare and contrast the related work to provide context for our research contributions. There are six related research areas that we describe: (a) stochastic hybrid models, (b) biochemical process models, (c) biochemical modeling based on hybrid systems, (d) simulation of stochastic systems, (e) verification of stochastic systems, and (f) analysis using Monte Carlo methods.

We present stochastic hybrid modeling paradigms to provide the context and justify our use of SHS through comparison with related formal modeling paradigms. We describe biochemical process modeling techniques to highlight previous efforts and identify the strengths and weaknesses of each approach. We also present biochemical models based on hybrid systems to compare our work with non-stochastic models.

Many analysis techniques exist, so we present related analysis methods for stochastic, hybrid, and biochemical systems to frame the contributions of our work. We consider related work in simulation of stochastic dynamical systems to justify our simulation methods and provide a context for our contributions. We also describe related verification techniques, discuss their limitations, and compare them with our methods. Lastly, we consider Monte Carlo-based analysis methods to provide a context for our approach on reachability analysis of SHS using Monte Carlo techniques.

Stochastic Hybrid Modeling Paradigms

Stochastic hybrid modeling paradigms are attractive for modeling biochemical systems because they combine continuous, discrete, and stochastic aspects in a formal context. However, several stochastic hybrid modeling paradigms exist, and they differ in the ways that they allow the incorporation of stochasticity. Some models allow the use of Stochastic Differential Equations (SDEs) to express the continuous dynamics, but others use Ordinary Differential Equations (ODEs). Two types of discrete dynamics can be found in stochas-

tic models: probabilistic transitions and deterministic transitions. The former determines when to fire the transition using a probabilistic distribution, while the latter utilizes guards and invariants to switch modes in the system deterministically. Both types are found in stochastic hybrid modeling paradigms, but each paradigm has different restrictions on the types of transitions that can be used.

Stochastic Hybrid Systems

General Stochastic Hybrid Systems In our work we use the General Stochastic Hybrid System (GSHS) model presented in [26]. GSHS combine stochastic, continuous, and discrete dynamics in a formal framework and utilize well-defined semantics for execution. Both probabilistic and deterministic transitions can be used for a GSHS model. The attributes of GSHS make it ideal for modeling biochemical systems that often require the combination of all the capabilities of GSHS.

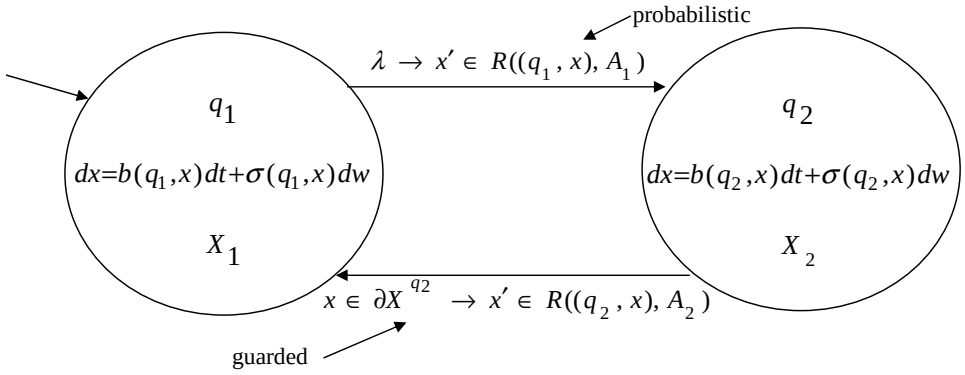


Figure 1: General stochastic hybrid system

Figure 1 shows a generic SHS model with two discrete states and two transitions (one probabilistic and one deterministic). The initial state of the system is q_1, and the continuous dynamics of the system evolve according to the SDE associated with q_1 until the probabilistic transition fires. The firing is determined by an exponentially decaying function defined by the firing rate λ [18]. Upon firing of a transition, the state resets according to the given reset map R. The state then evolves according to the SDE associated with q_2 until x crosses the boundary defined by ∂X^{q2}. A formal definition of GSHS and their execution are provided in Chapter III.

Stochastic Hybrid Systems SHS are introduced in [66] as a stochastic extension of hybrid systems using SDEs instead of ODEs. GSHS are introduced as an extension to SHS. SHS differ from GSHS in the fact that they lack probabilistic transitions. Only deterministic transitions are allowed between modes of the SHS restricting the types of systems that can be modeled by this paradigm. Figure 2 shows a diagram of an example SHS.

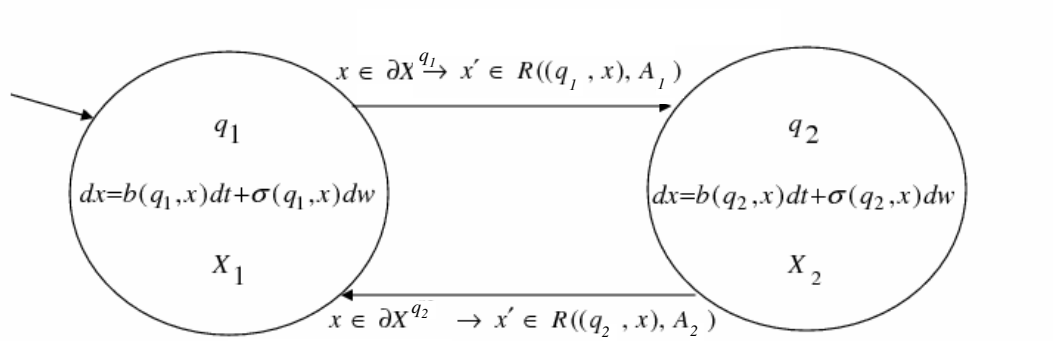


Figure 2: Stochastic hybrid system

The focus of [66] is on maximization or minimization of the reachability or safety probabilities for SHS. Embedded Markov chains are introduced by sampling from stochastic executions to aid in the calculation of the safety and reachability problems. The embedded Markov chain captures all the necessary details for safety and reachability analysis and is useful since explicit expressions for the execution of the SHS are impossible to obtain in general. Details on how to create the embedded Markov chain for any SHS can be found in [66]. The invariant distribution and exit probability from an interval of the Markov chain are studied, and it is shown that they converge to their counterparts for the solution to the original SDE as the limit of the discretization step goes to zero. Because SHS are a simplification of GSHS, virtually all of the analysis techniques for GSHS can be applied to SHS with only slight modifications.

SHS are used to model several important realistic systems. These include air traffic management [21], flexible manufacturing systems [48], biological systems [68], power management [65], and others.

Concurrent SHS Concurrent SHS (CSHS) are developed in [18] to incorporate inputs, outputs, hierarchy, and concurrency into SHS. The modeling tool Charon was modified to handle the stochastic dynamics of concurrent SHS. Charon utilizes a graphical user interface and allows parallel composition, instantiation, and information hiding that improve the readability of the models created. It also allows the use of shared variables for communication between the concurrent processes.

Charon includes a simulation engine that employs simple boundary crossing detection and the Euler-Maruyama method for SDE simulation. The modeling paradigm allows nondeterminism, so a stochastic mechanism for selecting from multiple transitions is also employed. Examples such as aircraft coordination and hard drive power management are modeled in Charon using concurrent SHS [18]. Figure 3 shows an example of a CSHS model of aircraft coordination with three aircraft and an air traffic controller. In this example the three aircraft models execute concurrently and the positions of the aircraft are shared variables.

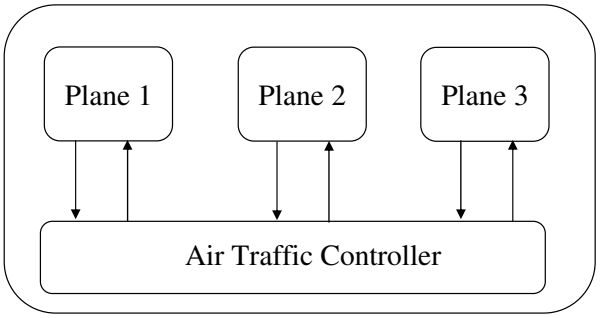


Figure 3: Concurrent SHS aircraft coordination example

Probabilistic transitions are not explicitly built into the language; however, they can be incorporated using the technique described in Chapter *IV*.

Deterministic Markov Processes

Piecewise Deterministic Markov Processes Piecewise Deterministic Markov Processes (PDMP) allow randomness in discrete transitions, but employ ODEs for the continuous dynamics. Similar to GSHS, PDMP utilize two types of autonomous transitions: discrete and probabilistic transitions. The dynamics are deterministic except for a sequence

of random transitions that can be defined at fixed or random times. Continuous time Markov chains can be thought of as a special class of PDMP; however, PDMP do not allow any diffusion in the continuous dynamics thereby limiting the types of systems that can be modeled.

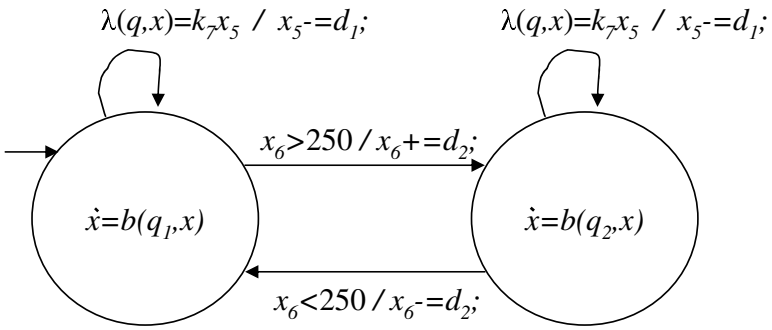


Figure 4: PDMP model of a biochemical system

PDMP are used to model queuing systems, storage inventory models, insurance mathematics models, and other systems [33]. Figure 4 shows an example of a PDMP biochemical model with two modes, two guarded transitions, and two probabilistic transitions. Reachability methods have been presented for PDMP in [25]. Because diffusion terms are not allowed, analysis techniques are simpler; however, many biochemical systems cannot be adequately modeled with PDMP.

Communicating PDMP Communicating Piecewise Deterministic Markov Processes (CPDMP) introduce automata to PDMP to make them compositional [131]. Defining composition in the formalism allows models to be compact for readability while retaining the inherent complexity of the model. CPDMP only allow ODEs for the continuous dynamics, so stochastic dynamics can only be introduced through discrete transitions. One-way synchronization is utilized between the concurrent components using 'passive transitions' where the active partner can influence the passive partner, but not vice versa. Passive transitions also occur independently of boundary-hit and probabilistic transitions.

Composition of CPDMP components is formally defined in [131]. Figure 5 shows an example of a CPDMP. Guarded, passive, and probabilistic transitions are respectively pictured as solid, dashed, and solid (with a box) arrows. Passive transitions are triggered by active transitions in a concurrent system. This model supports parallel components and

the execution engine handles the composition, which is typically exponential in size.

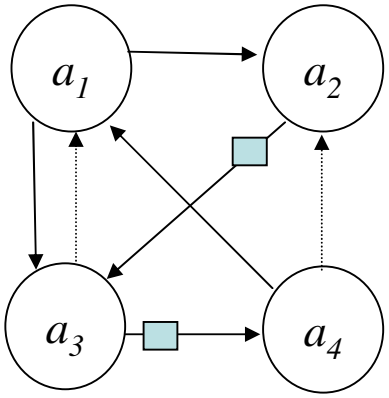


Figure 5: Composition of two CPDMP

CPDMP are used in examples of air traffic management [38]. CPDMP have been extended to accept multiple inputs and allow broadcasting to multiple components simultaneously. Future research has been proposed for extending the discrete communication to continuous interaction to increase the flexibility of the paradigm [132].

Related Models

Several related modeling paradigms utilize similar methods to describe stochastic dynamical systems.

Switching Diffusion Processes The Switching Diffusion Process (SDP) is a modeling paradigm that utilizes stochastic continuous dynamics and a controlled Markov chain to describe discrete jumps [47]. Both the continuous and discrete dynamics depend on the hybrid state of the system. SDP do not allow deterministic transitions or invariants, so the types of systems that they can model is limited. Fault tolerant control systems, failure-prone manufacturing systems, and multiple target tracking systems are some examples that have been modeled with SDP [47].

The focus of recent research on SDP has been on optimal control and stability. In [47] a non-random Markov policy that minimizes the pathwise long-run average cost is presented in the context of optimal control. In [97] schemes for stochastic stabilization

and destabilization with only partial control are presented. Continuing research is being conducted in improving efficiency and reducing complexity for SDP [97].

Jump Linear Stochastic Systems Jump Linear Stochastic Systems (JLSS) are found in the earliest stochastic hybrid literature and differ from SDP in the fact that they use a single differential equation to describe the dynamics instead of utilizing Markov chains. The equation that describes the evolution of the system is

$$dx_t = Ax_t dt + \sigma dw + Bu(t)dt + Rx_t dp_t$$

where the first two terms are from the standard SDE, $u(t)$ is an input signal, and p_t is a Poisson process. The input should be thought of as a disturbance that generates nondeterministic behavior rather than an external control input. Like SDP, JLSS do not allow deterministic transitions, which limits the types of systems that can be effectively modeled by them.

JLSS have been used to model target tracking, fault tolerant control, manufacturing processes, and other systems [109]. The earliest SHS models are based on JLSS since there is a large volume of work available on them. Most recent research has focused on controllability, stability, and optimal control [99], as well as bisimulation that can guarantee that two JLSS satisfy the same reachability or safety properties within a certain bound [74].

Taxonomy

In Table 1 we compare various modeling paradigms in the related literature. All of the modeling paradigms combine discrete, continuous, and stochastic aspects, but the way the stochasticity is incorporated varies. Concurrency is found in some models, but these are all extensions of previous modeling paradigms designed for enhanced usability.

Trade offs between the modeling paradigms motivate the use of one technique over another. Because our modeling contributions focus on biochemical systems, we choose to use GSHS because they support stochastic, continuous, and all types of discrete dynamics. The expressiveness of the language causes analysis methods to be computationally expensive; however, we demonstrate the feasibility of the simulation, verification, and Monte Carlo

Table 1: Modeling paradigm comparison

	SDE	ODE	Prob trans.	Guarded trans.	concurrency
GSHS	X		X	X	
SHS	X			X	
CSHS	X		X	X	X
PDMP		X	X	X	
CPDMP		X	X	X	X
SDP	X		X		
JLS	X		X		

techniques for realistic systems. From this point on in this chapter we refer to GSHS simply as SHS for simplicity.

Modeling Biochemical Processes

Biochemical systems can be inherently difficult or costly to analyze using traditional experimentation, so modeling and simulation methods have been developed to capture the unique nature of these systems. Biochemical models vary in the granularity of their focus as well as the complexity and types of interactions modeled. Some models incorporate stochastic dynamics, while others consider only deterministic interactions. Continuous and discrete dynamics can be used individually or combined within a model. We present the related biochemical modeling methods to provide the motivation and context for our work. The first set of modeling methods we present are coarser models that describe the interaction between components of a biochemical system. The second type of models we present are finer, dynamical models that capture the evolution of the individual aspects of a system over time.

Relational Models

Relational models are useful for describing rudimentary biochemical systems relations because they are easy to create, and analysis methods are simple to implement.

Directed and Undirected Graphs One of the simplest methods for modeling biochemical processes is a graph-based method where vertices correspond to genes and edges correspond to interactions between the genes. Directed graphs can be used to indicate inhi-

bition or promotion between pairs of genes. In Figure 6, gene A inhibits C and C promotes D.

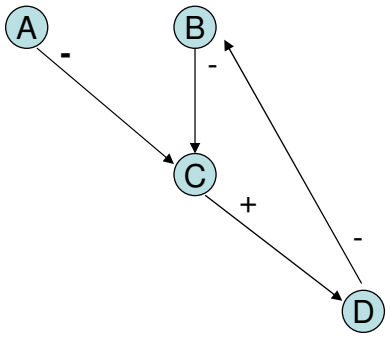


Figure 6: A directed gene graph example

This type of modeling is compact, easy to explore for large systems, and easy for predicting interactions. Furthermore, large databases exist that contain information about inhibition and promotion amongst genes. However, only a small number of operations can be performed on the graphs, and no quantitative analysis can be performed because of the simplicity of the model. It is possible to identify feedback loops by looking for cycles in the graph, and searching for paths between two genes can expose redundancy in the network [36].

Bayesian Networks A genetic regulatory system can be modeled using Bayesian networks as a directed acyclic graph where the vertices are genes (with corresponding random variables) and the edges imply direct regulators. The random variable corresponding to the vertices describe the expression level of that gene. In Figure 7 a simple Bayesian network is shown. This formalism has a solid basis in statistics that makes it attractive to capture the stochastic aspects of gene expressions [36].

Bayesian networks can also be used with an incomplete knowledge of the system, which is not necessarily true for other modeling methods. Artificial intelligence methods can be used to learn unknown interactions in Bayesian networks as well. Unfortunately, dynamical aspects of the regulatory system are left implicit, which can sometimes be overcome through generalizations in the modeling paradigm [44, 110].

Boolean Networks Boolean networks are based on elementary genetic principles and can be used model genes as being active (on) or inactive (off). A vector of 1s and 0s is

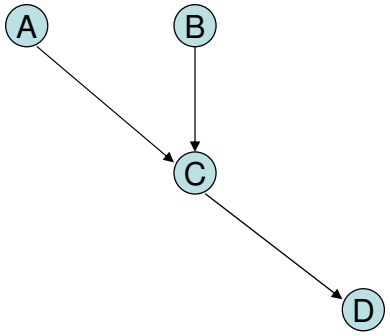


Figure 7: A Bayesian network example

sufficient to store the state of the system. Interactions between genes are expressed through boolean functions that define the next state of one gene from the current states of other genes. Boolean networks are efficient for revealing the behavior of large genetic networks by making strong simplifying assumptions about the dynamics and structure of the system [78]. One application of this method includes identification of suitable drug targets for cancer therapy [130].

Extensions to boolean networks to include stochasticity have been made to enhance their usefulness. In [130] it is shown that the properties that can be derived in a Bayesian network can be derived from a probabilistic boolean network as well. While these models are useful, they are fairly coarse-grained, so it can be difficult to discern precise relationships that can be discovered in finer models.

Generalized logical networks are another extension of boolean networks that allow more than binary values for genes. Logical variables are used as an abstraction of the actual concentrations of the genes in the system. Regulatory interactions are described by logical equations [134].

Dynamical Models

Dynamical models are useful for modeling systems that evolve over time. They are especially well suited for biochemical systems because chemical concentrations can be modeled using continuous and/or discrete dynamics in a formal way. Many methods exist, so we present several of the relevant paradigms to provide a context for our methods.

Nonlinear Ordinary Differential Equations Often, finer-grained modeling approaches

are necessary to capture the intricate behaviors of some biochemical systems. ODEs have been used extensively to model individual biochemical concentrations such as proteins, molecules, RNA, and other time-dependent variables. Interactions between the components are captured by the equations used to model the concentrations. Each equation contains a term for every influence on the concentration in the system, and since there can be many influences, these models can become very complicated, or the modeler must choose which interactions to include and which to leave out. Furthermore, the dynamics of each interaction can be challenging to accurately model because experimental results can be difficult to find, often requiring the use of estimated values [36].

Suppose we have a system of M chemical reactions and N chemical species. We define x_i as the concentration of the ith chemical species in micro-Molarity (μM), M_{fast} as the number of reactions, a_j as the reaction rate of the jth reaction, and the stoichiometric matrix v as a $(M_{fast} \times N)$ matrix whose values represent the number of chemical species lost or gained in each reaction. The dynamics for each of the i chemical species is described by

$$dx_i = \sum_{j=1}^{M_{fast}} v_{ji} a_j(x(t))dt \tag{1}$$

Due to the nonlinearity of the differential equations, analytical solutions are not generally available. Several integration methods exist that adequately handle ODE models. ODE models have been constructed for systems such as the lac operon in E. coli, the developmental cycle of bacteriophage T7, the expression of the HIV virus, circadian rhythms in Drosophilia, and others [36].

Discrete Stochastic Models Discrete models are a natural modeling paradigm for biochemical systems because they can capture the fine-grained changes of the concentrations of the involved reactants and products based on the stoichiometry defined by the biochemical reactions. In a discrete model, when the reaction fires, the concentrations of the reactants and products are reset to the appropriate updated values.

The rate at which chemical reactions occur is calculated using the stoichiometry defined by the type of reaction assuming temperature and pressure are constant. For example, the reaction $V + X \rightarrow Y + Z$, has a reaction rate $a = kvx$ where chemical species V, X,

Y, and Z have concentrations v, x, y, and z, respectively, and k is the reaction's kinetic coefficient. The rates of other types of reactions can be calculated similarly [62]. When the rate at which the reaction fires is determined by stoichiometry, the reactions can be simulated individually and the chemical species' concentrations updated individually [51].

This type of model is fairly easy to generate and highly accurate; however, because of the fineness of the approximation, simulation methods are very slow for large or complicated systems. A biochemical system drug model based on physical interactions at the molecular level has been developed in [118]. Additional examples can be found in [128].

Stochastic Master Equations The Stochastic Master Equation (SME) was derived in [51] to express the rate of change of each chemical concentration in a well-mixed solution. The rate of change of each chemical species is calculated using the chemical dynamics from the biochemical reactions. Suppose that we have a system of M chemical reactions and N chemical species. We define x_i as the concentration of the ith chemical species in micro-Molarity (μM), M_{fast} as the number of fast reactions, a_j as the reaction rate of the jth reaction, w as an M_{fast}-dimensional Wiener process, and the stoichiometric matrix v as a $(M_{fast} \times N)$ matrix whose values represent the concentration of chemical species lost or gained in each reaction. The dynamics for each of the i chemical species are described by

$$dx_i = \sum_{j=1}^{M_{fast}} v_{ji} a_j(x(t))dt + \sum_{j=1}^{M_{fast}} v_{ji}\sqrt{a_j(x(t))}dw_j. \tag{2}$$

This method is very effective for systems with large concentrations of chemicals, but if some concentrations are low, the model decreases in accuracy because infrequent interactions are not captured effectively. The addition of the stochastic term improves the accuracy of the model and does not significantly decrease the efficiency of the analysis methods.

A modeling technique that uses polynomial SHS to construct models for chemical reactions is presented in [62]. This technique utilizes a simplification of the SME formulation to model biochemical dynamics. A SHS model of a genetic regulatory network is compared to a deterministic model in [68]. The SHS model is derived from the specific dynamics of the genetic regulatory network and incorporates stochasticity using probabilistic transitions.

Fast/Slow Biochemical Reactions Discrete stochastic models such as the Stochastic

Simulation Algorithm (SSA) describe a reaction as firing at a rate calculated using the chemical concentrations and the kinetic coefficient. Slow reactions occur when reaction rates and concentrations are small enough and they can be modeled and simulated efficiently using discrete stochastic techniques. However, discrete simulations become inefficient when there are large concentrations of molecules and/or fast reaction rates. When discrete models become inefficient, reactions can be accurately modeled as continuous stochastic models using the chemical master equation [128].

Biochemical systems can contain a mixture of both fast and slow reactions. When fast and slow dynamics must both be considered it is most efficient to use a combined, hybrid modeling approach to take advantage of the efficiency of continuous modeling for the fast reactions while still keeping the accuracy of discrete modeling for the slow reactions. Determining which reactions are fast or slow is based on analysis of the rates using the kinetic coefficients and chemical concentrations. To determine the slowest rate, the smallest possible concentrations for each chemical species are used. Similarly, the fastest rate can be determined by using the highest possible concentrations. Since the reaction rates depend on the concentrations, reactions may be classified as either fast or slow dynamically based on the system state. Figure 8 shows an example of a fast/slow system of reactions in a SHS format with one slow transition. The fast dynamics are captured by the SDEs and the slow reaction is captured by the single self-loop transition.

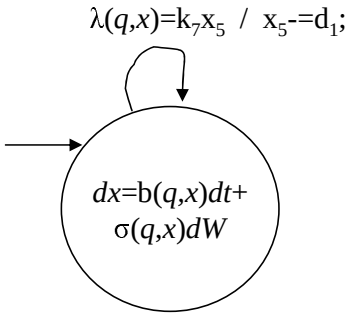


Figure 8: An example fast/slow model

Taxonomy

In Table 2 we compare the modeling techniques from this section. We consider the

resolution of the systems (coarse or fine), the discrete or continuous nature of the method, whether or not it incorporates stochasticity, and whether it is qualitative or quantitative.

Table 2: Modeling technique comparison

	Coarse/Fine	Discrete/Continuous	Stochastic	Qual/Quant
Directed graphs	Coarse	-	No	Qual
Bayesian Networks	Coarse	Discrete/Continuous	Yes	Quant
Boolean Networks	Coarse	Discrete	No	Qual
ODE	Fine	Continuous	No	Quant
SSA	Very Fine	Discrete	Yes	Quant
SME	Fine	Continuous	Yes	Quant
Fast/Slow	Fine	Discrete/Continuous	Yes	Quant

Each modeling method has advantages and disadvantages when compared to the other available methods. Coarse modeling methods are typically easier to generate and contain more global information; however, because of the lack of quantitative analysis methods, we do not consider them in our work. Both continuous and discrete methods inherently have advantages, so we consider both, and we also consider combinations of them. Because biochemical systems are inherently stochastic due to the uncertainty of molecular motion, our work focuses on stochastic systems.

Biochemical Modeling Based on Hybrid Systems

Hybrid systems are an appropriate modeling paradigm for biochemical systems because the systems often include continuous and discrete components. Furthermore, formal analysis techniques and tools are readily available for many classes of hybrid systems. A recently renewed interest in the field of biochemical hybrid system modeling has increased the quality and diversity of the models created.

Biological Protein Regulatory Networks

Biological protein regulatory networks have been modeled with linear hybrid systems in [50]. The presented modeling technique uses linear differential equations to describe the changes in protein concentrations and discrete switches to activate or deactivate the continuous dynamics based on protein thresholds. Symbolic parametric backward reachable

sets are computed from the equilibria of the hybrid system model and are guaranteed to be conservative underapproximations of the actual reachable sets. A multi-affine hybrid model of lactose metabolism is developed and analyzed with reachability analysis in [59].

The proposed technique is implemented in Matlab and results generated using the technique are presented [50]. The method is also applied to a piecewise affine hybrid system model of lateral inhibition through delta-notch signaling in [49]. A piecewise linear hybrid model for the lac operon used for regulating lactose metabolism in E. Coli is presented in [136] and studied in [39]. Hybrid systems are used in order to accurately model the behavior of excitable cells and cardiac tissue in [58, 138]. Our modeling technique uses similar switching dynamics to these techniques; however, we allow nonlinear, stochastic dynamics that can be used to improve the accuracy of the model.

Reactive Biological Systems

Two examples of hybrid system models of reactive biological systems are presented in [94]. Both systems model the concentration of the involved molecular species with continuous dynamics and other abstractions, approximations, and nonlinear effects with discrete dynamics. One models glucose metabolism and the other models the B. Subtilis sporulation inhibitor. Sigmoidal nonlinearities found in these real biological systems are modeled as piecewise linear functions to improve the accuracy over purely discrete models [94]. These modeling techniques are similar to our proposed method; however, our method allows stochastic dynamics and does not require nonlinear dynamics to be modeled with discrete transitions.

These models are implemented in HybridSAL [69]. The tool allows for safety analysis of the system using abstraction and a model checking algorithm. HybridSAL also allows for composition of multiple systems, which is ideal for creating large biological models. HybridSAL models can be abstracted into finite-state discrete transition systems automatically for simulation or verification [69].

Biomolecular Network Modeling

Biomolecular network modeling using hybrid systems is accomplished by using differen-

tial equations to model feedback mechanisms and discrete transitions to model changes in the underlying dynamics. Hybrid system models for the replessilator network and quorum sensing in bacteria are good examples because they highlight the benefits of using hybrid systems. Furthermore, the modeling tool Charon is presented in [7] along with the models to highlight the use of concurrency and communication often found in realistic models. Using hierarchical modeling tools such as Charon can help to aid in the creation of larger, more accurate models. Also, Charon supports stochastic simulation; however, it is not as accurate as our proposed simulation methods.

An abstraction method for medium-scale biomolecular networks using hybrid systems is presented in [59]. The method utilizes continuous multi-affine dynamics and allows for the automation of nonlinear rate laws that are not as accurate as our modeling techniques but allows for simpler verification. The abstraction method is useful for correctly identifying coarse, emergent features of the system and connecting them to the details of the underlying molecular dynamics. An example model of the lac operon is presented along with a method using reachability analysis that identifies parameter values using the steady state model structure. An approximation is constructed for an ODE model of the lac operon, and it is shown that the abstraction passes experimental tests that are used to test the original model. The system exhibits bistability and switching behavior arising from positive feedback in the expression mechanism of the lac operon [59].

Multi-Affine Hybrid Systems

Multi-affine hybrid systems with rectangular invariants are used to model phenomena such as quorum sensing [15], the stringent response [14], and lactose metabolism [59]. Reachability analysis for these multi-affine hybrid systems can be performed using the tool d/dt [15]. D/dt may not be able to perform reachability analysis in certain hybrid systems, but it is very efficient for some restricted cases of hybrid systems. In [14], it is shown that safety and reachability can be easily solved for a limited class of restricted systems. A multi-affine hybrid system model was created for analysis of bistability of the lactose induction system regulated by glucose and lactose [17]. A reachability method for multi-affine hybrid systems based on conical over approximation along with analysis results is also presented in [17].

Our modeling and verification method does not restrict the dynamics to affine but is less efficient for certain restricted systems.

Simulation of Stochastic Dynamical Systems

Simulation is a powerful analysis tool because simulation data can be directly compared to experimental data for validation of a model. Simulation data can also be useful in conditions where traditional experimentation is costly or impossible. Because biochemical system experimentation often has inherent disadvantages related to safety or cost, simulation is useful for these systems. Accurate and efficient simulation is important but challenging due to the inherent complexity found in biochemical systems.

Stochastic Simulation Algorithm

The Stochastic Simulation Algorithm (SSA) discretely simulates chemical reactions consuming reactants and creating products one reaction at a time. Probabilities of occurrence a_j are calculated for each reaction, and a stochastic sampling method is used to choose which reaction μ fires at each iteration as shown in Figure 9. Once a reaction fires, the quantities of reactants and products are updated [51]. The time step for the SSA can be determined using the following equation

$$\Delta t = \frac{1}{a_0} \ln \frac{1}{U}$$

where U is a uniformly-distributed number in $[0, 1]$.

The SSA is very accurate, but it can be inefficient for large systems or fast reactions because many iterations must be completed before results can be observed. To efficiently handle practical systems, computational improvements such as R-leaping have been devised for the SSA [12]. R-leaping increases the number of reactants consumed and products produced in each step by a factor of R. This increases the efficiency of the approximation, but decreases the accuracy as well. Because updates are made based on concentrations, the overall time step can vary throughout the simulation. An implementation of SSA can be found in the SimBiology package in Matlab [100]. We also develop our own implementation

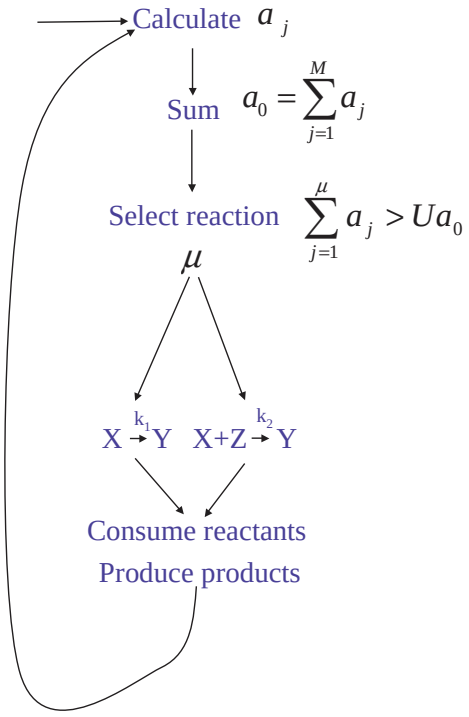


Figure 9: The stochastic simulation algorithm

of the SSA algorithm with R-leaping to use as a comparison with our simulation methods [120].

The Finite State Projection (FSP) algorithm can be applied to the solution of the chemical master equation as an alternative to the SSA algorithm and Monte Carlo methods [105]. The approach has been extended to systems with multiple time scales [106]. FSP utilizes a method of model reduction based on knowledge about a specific system. While this method is efficient, it is restricted to a finite-time horizon. Further, small changes to the model (such as medication effects) may change the model reduction. The lattice of states generated by the FSP could be used by our verification method and incorporating the model reduction technique into our SHS framework is a promising research direction.

Simulation of Stochastic Differential Equations

Simulation of SDEs can be performed using the Euler-Maruyama (EM) method, a first-order Taylor scheme [80]. The Milstein Method (MM) is a second-order Taylor scheme that

is more accurate while maintaining an acceptable efficiency. It extends the EM method by adding terms to the approximation; however, if the diffusion coefficient σ is constant, all higher order approximation terms are zero [80]. We describe the EM and MM methods in detail in Chapter IV.

ODE/SSA Simulation Methods

Several modeling and simulation techniques have been developed that combine other modeling techniques to improve the overall accuracy of the methods. Methods that combine the use of ODEs with the SSA are described in [6, 16, 79, 57]. A method that combines τ-leaping and the next reaction technique is described in [60]. These methods do not formally allow for the inclusion of external discrete dynamics, nor do they include stochastic continuous dynamics. Tools for the stochastic simulation of chemical reactions using these methods have been developed in [64, 4].

Variable Step Methods

Adjustable step integration methods are preferable to fixed step methods in general because efficiency and accuracy can potentially be improved through their use. However, variable step methods for stochastic systems are challenging because approximation of the Wiener process for variable steps is difficult, and exact characterization of the error at each step is impossible. Two types of variable step methods have been proposed: exponential time stepping and adaptive step methods [71, 90].

Simulation of SDEs can be performed using exponential time steps instead of fixed-size steps. The calculation of exponential time steps does not require Gaussian random variables, so the implementation is more efficient compared to fixed step methods. Furthermore, exponential boundary crossing tests can easily be performed between steps further increasing accuracy. No proof of convergence has been derived for the methods yet, and they are limited to multidimensional Brownian motion or single dimensional SDEs [71, 72].

Simulation of SDEs can also be performed using adaptive time step methods. These methods require an estimation of the error at each step to control the adaptive step size. Since the error cannot be explicitly calculated for most stochastic systems, general adaptive

algorithms do not exist. Furthermore, explicit calculation of the error bounds in general do not yet exist, so error estimation is required. While adaptive time step methods typically increase efficiency for systems with restricted dynamics, no order of convergence analysis is available [90].

Simulation of Stochastic Hybrid Systems

Simulation of SHS is challenging because it must accurately combine numerical integration methods for SDEs and detection/approximation of boundary crossings and reflections. Numerical integration of SDEs is accurate if the trajectory is sufficiently far from any boundaries; however, when the trajectory is close to a boundary, large errors can be incurred. Simulation methods for SHS have been developed for the modeling language Charon, but the focus is on concurrency, and the behavior close to the boundaries is not studied [18].

During the execution of a SHS, the process can hit a switching boundary defined by the invariants or guards. At a switching boundary the continuous process is halted and restarted in a new state after executing any transition resets. Switching boundaries can therefore be treated as absorbing boundaries. It is important to accurately estimate the time and location that the process is absorbed to minimize the error introduced into the approximation.

The easiest way to detect an absorbing boundary crossing is to check the state against the invariants at each step of the approximation. Let us assume the state at time t is $X(t)$. If $X(t) \in X^q$, but $X(t + \Delta t) \notin X^q$, then the process is rolled back to time t and restarted in the new state. An example of this is shown in Figure 10. In this example, the actual trajectory is shown by the dark line and actually crosses the boundary and returns without being detected. This type of error can be reduced by using smaller time steps. However, to avoid large error very small time steps may be required that could cause prohibitively slow computation [111].

An improved, stochastic method for absorbing boundary crossing detection based on stochastic sampling is developed in [56], and we present the details of this method in Chapter *IV*. A technique for accurately detecting absorbing boundaries has been developed for

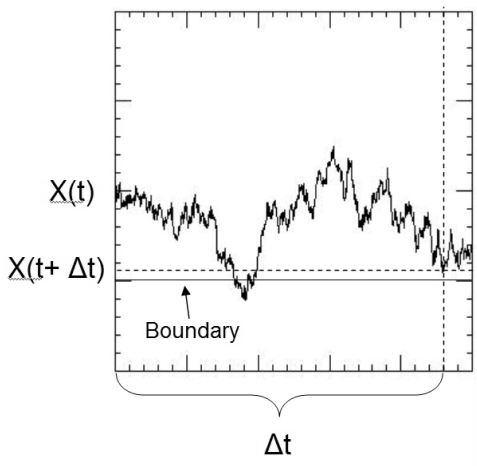


Figure 10: An example of an absorbing boundary

one-dimensional systems [96], and extensions have been proposed that scale to higher dimensional systems [91]. The boundary crossing detection algorithm presented in [111] uses analysis of moments to improve the accuracy of the approximation.

Reflecting boundaries are difficult to approximate because the crossing must be detected, and the reflection must be applied without incurring too much error. Approximating the crossing of the reflecting boundary incurs similar error to approximating the crossing of the absorbing boundary. Error generated in the approximation of the crossing is multiplied when calculating the reflection, so care must be taken when approximating the crossing. The traditional way to reflect the process is to detect the first crossing of the boundary and reverse the dynamics to force the system back into the valid bounds. This type of reflection may not be appropriate for the process, and care must be taken to ensure that the process is reflected into a valid state. Methods for approximating reflecting boundaries have also been studied previously [30]. We will present a stochastic methods for approximating reflecting boundaries from [56] in Chapter *IV*.

Modeling and Simulation Tools

Much development has gone into modeling and simulation software for biological systems because of the promise that these tools hold. One such tool Matlab's SimBiology, which allows modeling, designing, simulation, and analysis of biochemical pathways [100]. Another software package is SynBioSS that was developed as a tool for modeling biochemical systems

using the fast/slow method described previously. Further, SynBioSS is a repository for kinetic coefficients to aid in the development of large and complex models. It also provides a parallel implementation for improved efficiency with large systems [63]. The core of the simulation algorithm for SynBioSS is the Hy3S tool [129].

Taxonomy

We present simulation methods for stochastic dynamical systems to demonstrate the variability between and limitations of the available simulation methods. Methods range in simplicity from the most simple SSA algorithm to the complex SynBioSS software package. Efficiency of the methods is also highly variable from the efficient, but less accurate SDE methods to the highly accurate but less efficient SSA methods. Using a combination of SSA and ODE or SDE methods is effective for improving accuracy and efficiency, and further improvements can be made through formalization of the combination and by incorporating variable step methods. While the SHS simulation methods are the most flexible, accuracy and efficiency of these methods can be a challenge because of the inherent complexity of the interactions between the dynamics.

Verification

Exhaustive verification is an important task because it can take advantage of computational relationships in the model to provide an efficient, complete analysis. It is a challenging problem because of the inherent complexity of such a task, and systems with discrete and continuous dynamics further complicate the analysis of the dynamics. In this section we consider the verification of hybrid systems, stochastic systems, and SHS since these topics are related to our verification method.

Verification of Hybrid Systems

This section presents existing computational methods for (non-stochastic) hybrid systems. Hybrid systems can be verified by two types of techniques: *over-approximative* and *convergent* [135].

Over-Approximative Techniques In over-approximative verification techniques, each

step of the verification algorithm is designed to produce an over-approximation of the forward or backward reachable set. These methods use set representations such as polyhedra or ellipsoids and have been reported to scale well up to about six dimensions for general hybrid systems. If the reachable set is not initially found to be safe, it is required to tighten the verification variables and approximations. Therefore, multiple attempts may be necessary to verify a system, and it cannot be guaranteed that a solution can always be found.

The *d/dt* tool uses an over-approximation based on convex sets using griddy polyhedra expressed as closed-unit hypercubes with integer vertices [11]. Polyhedral approximations of flow pipes are used to calculate the forward reachable sets from an initial polyhedral set for linear dynamics. *CheckMate* is a Matlab-based tool that has similar requirements and uses a similar verification algorithm as *d/dt* [28]. *VeriSHIFT* is a tool that employs ellipsoidal sets and time-varying linear dynamics to calculate the reachable set using a similar technique as *d/dt* [22]. The *predicate abstraction* technique reported in [8] requires the specification of appropriate predicates that divide the state space into a finite number of regions. Continuous and discrete successors are calculated similarly to *d/dt*, but the predicate abstraction technique only calculates the intersection of the successors with other abstract states instead of the union with previous reachable states like *d/dt*, *CheckMate*, or *VeriSHIFT* [8].

Performance results have been reported in [40] for a navigation benchmark using *d/dt* and the predicate abstraction method. Because the performance of the verification using either technique varies with the choice of initial state, several initial states are tested. While some tests complete in just seconds, others are unable to return a solution. Some of the problems that *d/dt* could not verify are verified using predicate abstraction in as much as 78 minutes on an instance of the navigation benchmark with 9 discrete states. These tests were executed on a four processor Sun Enterprise 3000 with 4GB of memory.

Convergent Techniques Convergent approximative techniques solve the verification problem by approximating the hybrid system with another model of computation for which there exist well-understood verification methods. These techniques generally use grids to discretize the state space and allow the user to choose the resolution of the approximation. Examples of this technique include the Level Sets Method (LSM) [101] and our approxima-

tion using locally consistent discrete Markov processes.

The LSM is based on two-person, zero-sum game theory to determine an implicit representation of the boundary of the reachable set. Performance results are not as good as over-approximative methods, but the LSM has been used on systems with five dimensions [135]. One benefit of convergent techniques is that they generally do not restrict the dynamics of the system or the shape of the reachable set.

Verification of Stochastic Systems

Verification of stochastic systems is a difficult problem, so few methods exist. In [89], dynamics of state transitions are modeled as a discrete time Markov chain and an on-line model estimation method is presented as well as a goodness of fit test that statistically validates the model. A stochastic temporal logic called iLTL is created in order to specify aggregate behaviors of large scale systems. A method for verifying safety properties of hybrid systems is extended to include systems with stochastic dynamics in [115].

Stochastic π-calculus is a modeling framework that is able to express systems of concurrent components [88]. These components (or processes) each define a continuous time Markov chain, and therefore, they can be simulated using Gillespie's algorithm. Several tools have been created that implement the semantics of Gillespie's algorithm for stochastic π-calculus models [112, 116]. The models are countably infinite, so verification can only be performed symbolically, and current symbolic verification techniques are unable to handle nonlinear dynamics. Nonlinear dynamics are common in biochemical systems, so verification of realistic systems with this technique is limited.

Verification of SHS

A technique for stochastic verification for discrete-time SHS based on an optimal control formulation has been presented in [9]. The reachability problem for discrete-time SHS is formulated as a finite-horizon optimal control problem and is solved with a dynamic programming technique in [1], and is shown for both reachability and safety in [3]. An approximate dynamic programming approach for mitigating the curse of dimensionality when verifying SHS is presented in [2]. Computational methods based on theorem provers

for analyzing reachability of SHS based on the theory on Dirichlet forms have been presented in [24]. A method for safety verification based on over-approximation of the safe set using barrier certificates has been developed in [113].

SHS can be viewed as an extension of piecewise-deterministic processes [33] that incorporate stochastic continuous dynamics. Reachability of such systems has been studied in [25]. CPDMP have been presented in [131] with an emphasis on concurrency. Optimal control of piecewise deterministic processes has been studied in [34] where it is proved that the value function is the unique viscosity solution of a first-order Hamilton-Jacobi-Bellman equation. The work in [34] is based on a dynamic programming argument for characterizing the value function as a fixed point of an appropriate recursive operator and considers a discounted optimal control criterion that ensures that the recursive operator is contractive in order to prove convergence.

Reachability properties for continuous and hybrid systems have been characterized as viscosity solutions of variants of HJB equations in [95, 101]. Extensions of this approach to SHS and a toolbox based on level set methods have been presented in [102]. Level set methods are also based on a discretization of the state space and they offer computational advantages for many problems since the computation is limited to a neighborhood of the reachable set.

Discrete approximation methods based on finite differences have been studied extensively in [87] and the references therein. Convergence results justifying the use of discrete approximation techniques for stochastic optimal control problems have been presented in [43, 87]. Based on discrete approximations, the reachability problem can be solved using algorithms for discrete processes [117, 31, 35]. The approach has been applied for optimal control of SHS given a discounted cost criterion in [81]. For verification of reachability properties, the discount term cannot be used and convergence of the value function can be ensured only for appropriate initial conditions. A related grid-based method for safety analysis of stochastic systems with applications to air traffic management has been presented in [67].

Monte Carlo Methods for Reachability Analysis of SHS

Monte Carlo methods are useful for analyzing stochastic models as an efficient alternative to exhaustive verification methods. In this section we describe the solution of the reachability problem using Monte Carlo methods. Adaptive Monte Carlo methods and rare event detection methods are also presented as accuracy and efficiency improvement methods for traditional Monte Carlo methods.

Reachability Analysis Using Monte Carlo Methods

Reachability analysis for SHS can be performed using Monte Carlo methods because reachability can be described as the expectation of an indicator function [21]. If the indicator function describes the reachability of a state, then Monte Carlo analysis can be used to determine the reachability probability given an initial state [114]. Accuracy and efficiency can be tuned by adjusting the number of simulations. Stochastic roadmap simulation extends the Monte Carlo technique by analyzing multiple trajectories simultaneously to determine ensemble properties. The analysis of these ensemble properties can significantly improve the understanding of the entire system [10].

Monte Carlo analysis is performed by generating N trajectories from the same initial conditions with different Brownian motion and averaging the outcomes generated by the indicator function. The number of trajectories sampled directly affects the accuracy of the computed expected value. Because Monte Carlo techniques are based on sampling from simulations, error is introduced into the approximations [21].

Error is introduced into a Monte Carlo reachability analysis in two ways: time discretization error and statistical error. Time discretization error is introduced into the approximation through the integration method. Statistical error is introduced through the number of realizations of simulations that are used to determine the expected value [103].

To control the statistical error, it is necessary to change the number of Monte Carlo realizations. An accuracy can be achieved with a certain confidence by adjusting the number of independent extractions [114]. However, bounding the statistical error cannot guarantee the error of the entire approximation will be less than any value, so adaptive time stepping methods must be considered.

Adaptive Monte Carlo Methods

For certain systems it is important to bound the error of the approximation, so adaptive methods for switching diffusion processes have been created that adjust the time step size as well as the number of realizations to ensure that the error is less than a desired amount. The goal of adaptive time stepping algorithms is to choose the most efficient step size such that the error is less than a certain amount. Adaptive methods can use a priori or a posteriori knowledge at each time step to adjust the step size based on error estimation. Typically step sizes are adjusted by doubling or halving the step, and the step size is bounded to ensure that the solution is eventually found [103]. Care must be taken when approximating the Wiener process for a non-fixed time step, so generation algorithms for adaptive Wiener processes have been developed [90].

There are several different variable time stepping algorithms for approximating SDEs. In one algorithm, the step sizes vary in time, but are the same for all Monte Carlo samples [104]. Another algorithm uses step sizes that vary for different samples in the Monte Carlo approximation. The first method is useful for systems with small noise or singularities at deterministic times. The second method can be more efficient for certain systems and is better for systems with singularities at random times [104].

The biggest challenge of adaptive time stepping algorithms is the computation of the error estimation at each time step. A method for calculating the error along with examples is presented in [133]. This method is fairly efficient and can be used with both deterministic and stochastic variable time stepping methods. Another method for computing the error estimation uses a simpler approach based on the derivative of the trajectory, but it does not include any proven error bounds, unlike other methods [129].

Rare Event Detection

Certain models of systems such as air traffic management or health care require very strict error bounds that can require prohibitively expensive computation using Monte Carlo methods. Rare events such as entering an unsafe region of the state space are difficult to detect for most systems. For some specific problems techniques have been developed that

isolate rare events and calculate their impact on the system more efficiently [86].

One general solution to this problem is to characterize the expected value of the reachability probability as a product of conditional probabilities of intermediate events leading to it. This type of analysis requires significant domain knowledge and cannot provide absolute error guarantees, but it can provide significantly more efficient simulations for these rare event scenarios [86].

Importance sampling is a rare event detection method that requires the adjustment of the probability laws that drive the system to increase the sampling near the rare event [55, 23]. Importance sampling has been used for switching diffusions [86] and a large SHS model in [20], and the main challenge is determining appropriate ways to adjust the probability laws. Much research work has focused on improving importance sampling; however, each system must be analyzed individually to determine the appropriate change of measure for the dynamics to reduce the variance. If the incorrect change of measure is utilized, the results may turn out worse than if traditional Monte Carlo methods are used [53].

MultiLevel Splitting (MLS) is more effective for many systems and easier to implement than importance sampling, and is therefore the method that we present in detail in a later section. The original concept of splitting is presented in [76]. Previous MLS algorithms have been applied to SDEs [93], but to our knowledge have not been applied to SHS. MLS has two main implementations: fixed-splitting and fixed-effort. Fixed-splitting methods require a fixed value for the number of trajectories to split at each level, and can be performed in a depth-first manner. Fixed-effort methods choose a maximum number of trajectories to be split at each level and divide the split trajectories at each level in a breadth-first approach. Fixed-effort requires more memory and processing but can have improved performance for certain systems. Fixed-splitting is faster, more efficient, and simpler to tune, so it is the method that we use in our work [93].

A MLS splitting policy has been proposed to replace the boundaries and splitting levels with a splitting function. The evaluation of the function determines the appropriate time and number of forks for splitting the trajectory. This method allows the use of any type of function, so it is a generalization of the MLS method [93].

In order to reduce the variance optimally, it has been shown that trajectories with

the highest probability of leading to the region of interest should be split [54]. The main challenge of MLS methods is placing boundaries and choosing splitting policies or functions in the best way [93].

Comparison with Related Work

The goal of this dissertation is to develop computationally efficient, accurate methods for modeling and analyzing biochemical systems. The related work presented in this chapter is intended to provide a context for the contributions of our work. Our contributions extend previous work in three major areas: modeling, simulation, and verification.

Modeling

While relational biochemical models [36, 44, 110, 78, 134] are useful for fast prototyping, they cannot be used for quantitative analysis, so we use a dynamical modeling paradigm. Discrete dynamical models can be very accurate, but are also very inefficient to analyze, and continuous models are efficient, but can be inaccurate. Our modeling approach allows both discrete and continuous dynamics to take advantage of the strengths and avoid the weaknesses of each method. Our approach also allows easy inclusion of realistic dynamics such as drug interactions or temperature fluctuations, which previous models did not include.

Our modeling approach uses the framework of SHS to formalize and extend similar previous modeling methods [128]. We use GSHS because they allow for continuous and discrete dynamics in the a general, formal, and stochastic framework. Most other modeling approaches combine some of these attributes, but not all of them (as seen in Tables 1 and 2). Concurrency and hierarchy are not part of the formal SHS definition, but they could be added easily to our modeling tools as in [18].

Modeling the effect of drugs and specifically enzyme inhibitors, is an important task because it can enhance the understanding prior to the execution of actual experiments that have inherent disadvantages. Previous drug modeling has focused mainly on the physical interactions at the molecular level [118, 5, 42]. Modeling of enzyme inhibitors has not, to our knowledge been attempted in conjunction with chemical reactions.

We present a modeling methodology that can be used to model a large class of biochemical systems, and we also present realistic case studies to demonstrate the methodology. While modeling methodologies have been previously presented [128], only small, simple examples are presented to demonstrate the results. Further, we present experimental results from our case studies to demonstrate the realistic usability of our methodology.

Simulation

Accurate simulation of dynamical models is a challenge that has been addressed previously [80, 71, 72, 90, 111, 30], but has not been extended for SHS. We take the previous results, combine and improve them to create an improved comprehensive SHS simulation algorithm.

A simulation engine for SHS has been developed for the modeling tool Charon [18], but it is based on simple numerical integration and boundary crossing detection methods, which limits its accuracy and efficiency. In this dissertation we develop an advanced simulation technique for SHS that employs improved boundary crossing detection methods for absorbing and reflecting boundaries utilizing probabilistic methods to improve both accuracy and efficiency.

Previous methods that utilize adaptive time stepping do not consider discrete dynamics [90, 71, 72]. We incorporate the adaptive time stepping ideas from these methods into our SHS simulation algorithms to take advantage of our improved methods and avoid error that can be caused by discrete transitions.

Verification

Our work in verification considers a reachability criterion for SHS for which the value function is a viscosity solution of a set of coupled second-order Hamilton-Jacobi-Bellman equations. The results of [34] cannot be applied because the presence of stochastic continuous dynamics and the absence of a discount factor in the reachability criterion require new techniques for proving convergence. The dynamic programming approach that we use is generally simpler to implement and captures the dependency of the value function between discrete modes. The approach also allows us to show the convergence of the solution

obtained using the numerical methods to the solution of the SHS.

It is difficult to compare our verification performance results with other approaches because the performance of our algorithm is not dependent on a set of initial states unlike most over-approximation techniques. Furthermore, our technique verifies SHS with nonlinear continuous dynamics. We have verified a stochastic version of the collision avoidance problem described in [135], and we found that our method has similar performance to the LSM. One of the advantages of our method is that it is easily parallelized by applying known decomposition methods from parallel dynamic programming [19].

Monte Carlo methods for reachability analysis of SHS have been proposed previously [114]. The use of adaptive error control methods for Monte Carlo simulation of SDP have been proposed as well [103]. Our work utilizes the higher order simulation techniques with adaptive time stepping and improved boundary crossing detection to create a more accurate Monte Carlo simulation engine for SHS.

Variance reduction methods for SHS have been previously presented [86, 20]; however, these methods use importance sampling or conditional probability methods that can be complex to implement and require a detailed understanding of the dynamics. MLS has been presented previously [93], but it has not been used for systems with discrete dynamics. We also present practical methods for determining parameters for MLS along with experimental results to demonstrate tangible improvements in both accuracy and efficiency.

CHAPTER III

MODELING OF BIOCHEMICAL SYSTEMS USING SHS

As biomedical research advances into more complex systems, there is an increasing demand for accurate, extensible modeling methods specifically suited to biochemical systems. Models of biochemical processes can provide insight into the modeled systems and allow development and testing of hypotheses that may not be possible to test in the real system because of physical or cost limitations. Further, models can be extended, compared, and refined to increase their long term utility.

Modeling for biochemical systems is challenging because the systems are typically highly interconnected and complex. Most current biochemical modeling methods are either too simple for realistic problems or too complicated for domain experts to easily adopt. Further, models are often custom generated and difficult to enhance or change by the domain expert. Also, few biochemical models are based in formal methods, so it can be difficult to formally analyze system properties.

We present a biochemical system modeling framework using Stochastic Hybrid Systems (SHS). Because biochemical processes are inherently stochastic and often contain both continuous and discrete behavior, SHS capture their combined dynamics effectively. We present a method for modeling the effect of temperature on the reactions as well as deterministic and stochastic methods for modeling medication administration. These modeling methods provide a framework for modeling complex, realistic models.

In this chapter we present several realistic models to demonstrate the flexibility and usability of our methodology with realistic systems. Each model is presented to demonstrate a different aspect of our modeling methodology. The systems range in size from three continuous dimensions to 22.

We present three sugar cataract development models that have varying medication administration policies to demonstrate the flexibility and ease of modeling medications [121, 120, 122, 124]. The first model describes the biochemical process of sugar cataract development without the presence of medication. The two subsequent models extend the

first model to include the effect of medication on the system. The first medicated model assumes that the effect of the drug on the system is instantaneous, while the second medicated model is designed to incorporate stochastic delay to model absorption and metabolization.

We describe the biodiesel production process, establish realistic parameters for its simulation, and we present a SHS model for the biodiesel processor and reactions. Our model incorporates temperature fluctuations due to a thermostat-controlled heater and models the resulting effects on the chemical reaction kinetics. It also incorporates glycerol separation, a process used to increase product quality in real biodiesel production systems [119, 126]. Biodiesel is a well-studied process that offers many experimental data sets, so we present validation of the model using experimental data in Chapter *IV*.

Glycolysis is a biochemical process found in virtually every living cell that converts sugars into usable energy molecules for the cell [125]. We present a model of glycolysis that incorporates a feeding mechanism. The continuous state space has 22 dimensions, and there are two discrete modes to describe the feeding process. We present this model because it is a large, yet well developed system that demonstrates the limitations of existing analysis methods.

The water balance problem in humans is regulated by a robust system that we model to demonstrate the ability of SHS to capture macro-scale behavior in a realistic system [123]. All organisms must regulate the balance of water and electrolytes in their bodies. In humans, this regulation is part of a system that involves the kidneys, circulatory system, pituitary gland, and other minor components. Certain aspects of the water balance problem exhibit challenges for accurate simulation, so we present this model to demonstrate the ability of SHS to accurately model these types of behavior.

The rest of the chapter presents the formal SHS model followed by our SHS modeling methodology for biochemical systems and several example case studies and SHS models. The first case studies are of the sugar cataract development process using multiple medication administration policies, the next model is the biodiesel production process, followed by the glycolysis system and the water/electrolyte balance system. We conclude the chapter with a summary of the work.

Stochastic Hybrid Systems

In our work we use the General Stochastic Hybrid System model presented in [26]. GSHS combine stochastic, continuous, and discrete dynamics in a formal framework and utilize a well-defined semantics for execution. These attributes make it ideal for modeling biochemical systems that often require the combination of all the capabilities of GSHS.

To establish the notation we let Q be a set of discrete states. For each $q \in Q$, we consider the Euclidean space $\mathbb{R}^{d(q)}$ with dimension $d(q)$ and we define an *invariant* as an open set $X^q \subseteq \mathbb{R}^{d(q)}$. The hybrid state space is denoted as $S = \bigcup_{q \in Q} \{q\} \times X^q$. Let $\bar{S} = S \cup \partial S$ and $\partial S = \bigcup_{q \in Q} \{q\} \times \partial X^q$ denote the completion and the boundary of S respectively. The Borel σ-field in S is denoted as $\mathcal{B}(S)$.

Definition 1 *A GSHS is defined as* $H = ((Q, d, \mathcal{X}), b, \sigma, Init, \lambda, R)$ *where*

- Q *is a set of discrete states (modes),*
- $d : Q \to \mathbb{N}$ *is a map that defines the continuous state space dimension for each* $q \in Q$*,*
- $\mathcal{X} : Q \to \mathbb{R}^{d(\cdot)}$ *is a map that describes the invariant for each* $q \in Q$ *as an open set* $X^q \subseteq \mathbb{R}^{d(q)}$*,*
- $b : Q \times X^q \to \mathbb{R}^{d(q)}$ *and* $\sigma : Q \times X^q \to \mathbb{R}^{d(q) \times p}$ *are drift vectors and dispersion matrices respectively,*
- $Init : \mathcal{B}(S) \to [0, 1]$ *is an initial probability measure on* S*,*
- $\lambda : \bar{S} \to \mathbb{R}_+$ *is a nonnegative transition rate function, and*
- $R : \bar{S} \times \mathcal{B}(\bar{S}) \to [0, 1]$ *is a transition measure.*

To define the execution of the system, we denote $(\Omega, \mathcal{F}, P)$ the underlying probability space, and consider an $\mathbb{R}^p$-valued Wiener process $w(t)$ and a sequence of *stopping times* $\{t_0 = 0, t_1, t_2, \ldots\}$. Let the state at time t_i be $s(t_i) = (q(t_i), x(t_i))$ [1] with $x(t_i) \in X^{q(t_i)}$. While

[1] When there is no confusion, we use interchangeably the notation (q, x) and s for the hybrid state to simplify complex formulas and often we use the notation $s_{t_i} = (q_{t_i}, x_{t_i})$ for brevity.

the continuous state stays in $X^{q(t_i)}$, $x(t)$ evolves according to the Stochastic Differential Equation (SDE)

$$dx = b(q, x)dt + \sigma(q, x)dw \tag{3}$$

where the discrete state $q(t) = q(t_i)$ remains constant and the solution of (3) is understood using the Itô stochastic integral [73]. A sample path of the stochastic process is denoted by $x_t(\omega), t > t_i, \omega \in \Omega$. The next stopping time t_{i+1} represents the time when the system transitions to a new discrete state. The discrete transition occurs either because the continuous state x exits the invariant $X^{q(t_i)}$ of the discrete state $q(t_i)$ or based on an exponential distribution with transition rate function λ. Therefore, t_{i+1} can be defined as the minimum between two other stopping times: (i) The first hitting time of the boundary $\partial X^{q(t_i)}$ defined as $t^*_{i+1} = \inf\{t \geq t_i, x(t) \in \partial X^{q(t_i)}\}$ and (ii) a stopping time τ_{i+1} described by an exponential distribution with survivor function

$$M(t, \omega) = \exp\left(-\int_{t_i}^{t} \lambda(q(t_i), x_z(\omega))dz,\right),\ \omega \in \Omega.$$

Thus, the time of the next discrete transition t_{i+1} is a stopping time whose distribution is defined by the survivor function

$$F(t, \omega) = I_{(t<t^*_{i+1})} \exp\left(-\int_{t_i}^{t} \lambda(q(t_i), x_z(\omega))dz\right)$$

where I denotes the indicator function. [2]

At time t_{i+1} the system transitions to a new discrete state and the continuous state may jump according to the reset measure R. The trajectory of $x(t)$ is assumed to be left-continuous, so we denote $x(t^-_{i+1})$ the solution of (3) at $t = t_{i+1}$ and $s(t^-_{i+1}) = (q(t^-_{i+1}), x(t^-_{i+1}))$ where $q(t^-_{i+1}) = q(t_i)$ the discrete state before the transition. If $t_{i+1} = \infty$, the system continues to evolve according to (3) with $q(t) = q(t_i)$. If $t_{i+1} < \infty$, the system jumps at t_{i+1} to a new state $s(t_{i+1}) = (q(t_{i+1}), x(t_{i+1}))$ according to the transition measure $R(s(t^-_{i+1}), A)$ with $A \in \mathcal{B}(S)$. The evolution of the system is then governed by the SDE (3) with $q(t) = q(t_{i+1})$ until the next stopping time.

[2] Given a set $A \in \mathcal{F}$ the indicator function is defined as $I_A(\omega) = 1$ if $\omega \in A$ and 0 if $\omega \notin A$.

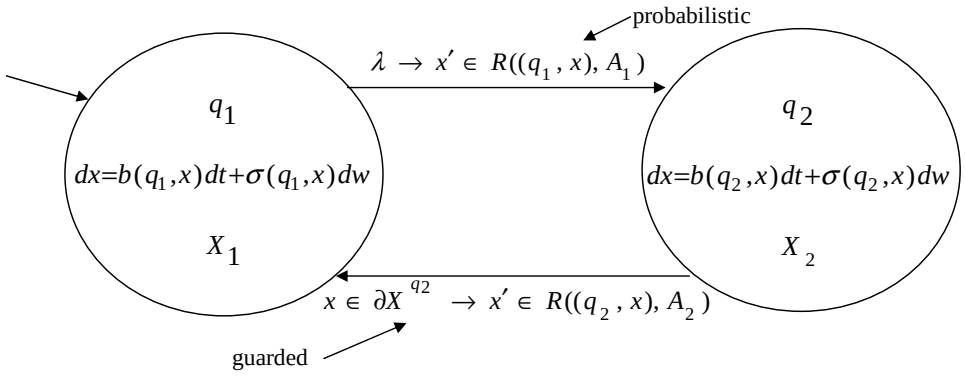


Figure 11: General stochastic hybrid system model

Figure 11 shows a generic GSHS model with two states and two transitions (one probabilistic and one deterministic). The continuous dynamics of each state are defined by the associated SDE. The probabilistic transition fires at the firing rate λ, and the guarded transition fires when x hits the boundary $x \in \partial X^{q_2}$. The logical condition $x \in \partial X^{q_2}$ is often referred to as the guard of the transition. Upon firing of a transition, the state resets according to the map $R((q,x),A)$.

The following assumptions are imposed on the model. The functions $b(q,x)$ and $\sigma(q,x)$ are bounded and Lipschitz continuous in x for every q, and thus the SDE (3) has a unique solution. The transition rate function λ is a bounded and measurable function that is assumed to be integrable for every $x_t(\omega)$. For the transition measure, it is assumed that $R(\cdot, A)$ is measurable for all $A \in \mathcal{B}(S)$, $R(s,\cdot)$ is a probability measure for all $s \in \bar{S}$, and $R((q,x),dz)$ is a stochastic continuous kernel. An important characteristic of the model used in our analysis is that it satisfies the strong Markov property [26].

Let $N_t = \sum_i I_{t \geq t_i}$ denote the number of jumps in the interval $[0,t]$. It is assumed that the expected number of jumps is finite for every initial state $s \in S$, that is $E_s[N_t] < \infty$. A sufficient condition for ensuring finitely many jumps can be formulated by imposing restrictions on the transition measure $R(s,A)$. Let $s = (q,x)$ be the state after a discrete transition. If for every $x \in A$, $d(x, \partial X^q) \geq \epsilon > 0$ [3] and $\exists \delta > 0$ such that $P[\inf\{t > t_{i+1}, x(t) \in \partial X^q\} \geq \delta] = 1$ then $t_{i+1} - t_i > \delta, i = 1, 2, \ldots$ with probability 1. This condition is satisfied if the continuous state after a jump is in the interior of an invariant. For the

[3] $d(x, \partial X^q)$ denotes the Euclidean distance between x and ∂X^q

rest of this dissertation we refer to GSHS as SHS for simplicity.

Modeling Methodology

All cellular function of living organisms is governed by complex systems of coupled biochemical reactions. A reaction specifies all chemical species that react (reactants) and are produced (products). A kinetic constant k associated with each reaction quantifies the affinity for the reactants to produce the products in defined temperature and pressure conditions.

Experimental analysis can be used to physically measure the variation in individual concentrations of the chemical species in a biochemical system. However, understanding the dynamical behavior of biochemical systems requires running many experiments that can be time consuming, tedious, prone to error, unsafe, or costly. Developing and analyzing dynamical models for capturing the evolution of individual chemical species concentrations can reduce the number of experiments needed and accelerate the understanding of the system.

Discrete models are a natural modeling paradigm for biochemical systems because reactions can be considered as occurring at specific points in time, and when a reaction occurs, individual molecules interact and produce new molecules. Discrete models update the concentrations of the involved reactants and products at a certain reaction rate based on the stoichiometry defined by the reaction.

Chemical reactions are inherently stochastic because of the unpredictability of molecular motion [37], so their dynamics are best described by stochastic models. Discrete stochastic models of reactions can be created by describing a reaction j as firing at a rate a_j [27]. When the reaction fires, the concentrations of the reactants and products are reset to the appropriate updated values. Table 3 shows the rates and resets for several examples of different types of reactions. For example, when the reaction $X \rightarrow Z$ occurs, a molecule of X is consumed and a molecule of Z is produced denoted by $x- = 1$ and $z+ = 1$ respectively, where x and z are the quantities of molecules of chemical species X and Z, and k_i is the kinetic constant for reaction i.

Reactions occur at different speeds depending on the concentrations of chemicals and

Table 3: Example reactions, reaction rates, and resets

Reaction	a_j	Reset
$X \rightarrow Z$	$k_1 x$	$x- = 1;\ z+ = 1;$
$X + Y \rightarrow 2Z$	$k_2 xy$	$x- = 1;\ y- = 1\ ;\ z+ = 2;$
$2X \rightarrow Z$	$1/2 * k_3 x(x-1)$	$x- = 2;\ z+ = 1;$
$2X + Y \rightarrow 2Z$	$1/2 * k_4 x(x-1)y$	$x- = 2;\ y- = 1;\ z+ = 2;$
$3X \rightarrow Z$	$1/6 * k_5 x(x-1)(x-2)$	$x- = 3;\ z+ = 1;$

the kinetic constant for each reaction. 'Slow' reactions occur when reaction rates and concentrations are small enough and they can be modeled and simulated efficiently using discrete stochastic techniques. However, discrete simulations become inefficient when there are large concentrations of molecules and/or fast reaction rates. In such cases the reaction occurs very frequently and the discrete simulation needs to consider a large number of transitions in a short period of time. "Fast" reactions occur at a rate that is fast enough or in high enough concentrations to consider as occurring at a constant rate. Such reactions can be modeled more efficiently as stochastic continuous models assuming the reactions occur in a well-mixed solution [128].

The rate of change of each chemical species is calculated using the chemical dynamics from the biochemical reactions. Suppose that we have a system of M chemical reactions and N chemical species. We define x_i as the concentration of the ith chemical species in micro-Molarity (μM), M_{fast} as the number of fast reactions, a_j as the reaction propensity of the jth reaction, and W as an M_{fast}-dimensional Wiener process. The stoichiometric matrix v is a (M_{fast} X N) matrix that holds values representing the concentration of chemical species lost or gained in each reaction. The following equation describes the dynamics for each of the i chemical species [128].

$$dx_i = \sum_{j=1}^{M_{fast}} v_{ji} a_j(x(t))dt + \sum_{j=1}^{M_{fast}} v_{ji} \sqrt{a_j(x(t))} dW_j \tag{4}$$

Discrete and continuous models consider only slow or only fast chemical reactions, but real biochemical systems often contain a mixture of both fast and slow reactions. In such a situation it is most efficient to use a hybrid modeling approach to take advantage of the

efficiency of continuous modeling while still keeping the accuracy of discrete modeling [128].

Stochastic hybrid systems are ideal for modeling biochemical systems with both fast and slow chemical reactions systems because they are able to model continuous and discrete dynamics in a stochastic framework. Fast reactions are modeled using the continuous stochastic dynamics techniques presented earlier, and slow reactions are modeled as discrete transitions with stochastic rates and resets.

To determine which reactions are fast or slow, one must analyze the rates using the kinetic parameters and quantities of each reactant involved. The reaction rate range can be determined by analyzing the rate a_j from Table 3 over the entire range of possible chemical concentrations. To determine the smallest rate, the smallest concentrations for each chemical species should be used. Similarly, the largest rate can be determined by using the highest concentrations in the range. Reactions can be considered slow if the reaction rate never exceeds 100 reactions per second, otherwise reactions can be modeled as fast reactions. If a reaction has a range that spans 100 reactions per second, the reaction can be classified as either fast or slow.

All chemical reaction rates are effected by the temperature at which they occur. The higher the temperature, the more likely that the individual molecules will interact and eventually react. The chemical reaction rate k is most often defined for a single temperature and pressure, but most chemical reactions are exothermic or endothermic and therefore inherently change the temperature.

Furthermore, it is advantageous to control the reaction rates by applying or removing heat to ensure that the system behaves correctly. The effect of temperature on the reaction rate, k, is given by $k = Ae^{\frac{-Ea}{RT}}$ where A is a constant for each reaction, Ea is the activation energy for each reaction, R is the gas constant (1.9872), and T is the temperature in Kelvin (for example see [32]). Using this equation we can determine the reaction rates for each reaction at any temperature and therefore model the fluctuating reaction speeds.

A heating (or cooling) apparatus generally heats or cools in a binary manner (on or off), so a discrete model of heating control is necessary. Temperature can easily be included in a stochastic model as another continuous state. The temperature can then be used to help calculate the reaction rates for the individual reactions.

Sugar Cataract Development

This section describes three SHS models of the biochemical process of Sugar Cataract Development (SCD). The first model describes the biochemical process of SCD. The two subsequent models extend the first model to include the effect of medication on the system. The medicated model we present first assumes that the effect of the drug on the system is instantaneous, while the second medicated model incorporates stochastic delay to model absorption and metabolization.

These models are not only used to demonstrate the modeling methodology, but also our analysis methods. In the Chapter IV we present simulation trajectories of the SCD models, and we validate the SCD models using the SSA algorithm. In Chapter V we present verification analysis of the three SCD models.

SHS Sugar Cataract Development Model (SCD1)

A sugar cataract distorts the light passing through the lens of an eye by attracting water to the lens when an excess of sorbitol is present. Often these cataracts are formed in the eyes of diabetic patients who have highly fluctuating blood sugar levels. Several factors affect the accumulation of sorbitol including the amount of the enzyme SDH. SDH catalyzes the reversible oxidation of sorbitol and other polyalcohols to the corresponding keto-sugars [98]. There are 8 chemical species involved in the reaction: $NADH($ $x_1)$, $E-NADH$ (x_2), NAD^+ (x_3), $E-NAD^+$ (x_4), SDH (x_5), Fructose (x_6), Sorbitol (x_7), and the inactive form of SDH (Z).

A SHS model for SCD (SCD1) has been previously presented in [128, 120]. The bovine lens data is used as a standard model for human cataract development. The ranges are bounded and are estimated using realistic concentration values derived from experimental data and Michaelis-Menten constants (Km) defined as the rate of the reaction at half-maximal velocity [98]. Table 4 describes the seven reactions and rates involved in SCD. The rates are calculated based on the average concentrations for each chemical species and the kinetic coefficients presented in Table 4. The first six reactions are classified as fast and the seventh is classified as slow because it is significantly slower than the other reactions.

Table 4: Sugar cataract reactions and kinetic coefficients

Reaction	Kinetic coefficient	Rate
$SDH + NADH \rightarrow E - NADH$	$k_1 = 6.2$	31.1
$E - NADH \rightarrow SDH + NADH$	$k_2 = 33$	151
$E - NADH + F \rightarrow E - NAD^+ + S$	$k_3 = 0.0022$	6
$E - NAD^+ + S \rightarrow E - NADH + F$	$k_4 = 0.0079$	19.5
$E - NAD^+ \rightarrow SDH + NAD^+$	$k_5 = 227$	998
$SDH + NAD^+ \rightarrow E - NAD^+$	$k_6 = .61$	3.2
$SDH \rightarrow Z$	$k_7 = 0.0019$	0.002

Each of the six fast reactions are modeled using the SDE (4). The inactive form of SDH (Z) is not a reactant in any of the chemical equations, so its concentration is not modeled. The equations describe the rates of change of the individual chemical species and are

$$\begin{aligned}
dx_1 &= (-k_1x_1x_5 + k_2x_2)dt - \sqrt{k_1x_1x_5}dw_1 + \sqrt{k_2x_2}dw_2 \\
dx_2 &= (k_1x_1x_5 - k_2x_2 - k_3x_2x_6 + k_4x_4x_7)dt + \sqrt{k_1x_1x_5}dw_1 \\
&\quad -\sqrt{k_2x_2}dw_2 - \sqrt{k_3x_2x_6}dw_3 + \sqrt{k_4x_4x_7}dw_4 \\
dx_3 &= (k_5x_4 - k_6x_3x_5)dt + \sqrt{k_5x_4}dw_5 - \sqrt{k_6x_3x_5}dw_6 \\
dx_4 &= (k_3x_2x_6 - k_4x_4x_7 - k_5x_4 + k_6x_3x_5)dt + \sqrt{k_3x_2x_6}dw_3 \\
&\quad -\sqrt{k_4x_4x_7}dw_4 - \sqrt{k_5x_4}dw_5 + \sqrt{k_6x_3x_5}dw_6 \\
dx_5 &= (-k_1x_1x_5 + k_2x_2 + k_5x_4 - k_6x_3x_5)dt - \sqrt{k_1x_1x_5}dw_1 \\
&\quad +\sqrt{k_2x_2}dw_2 + \sqrt{k_5x_4}dw_5 - \sqrt{k_6x_3x_5}dw_6 \\
dx_6 &= (-k_3x_2x_6 + k_4x_4x_7)dt - \sqrt{k_3x_2x_6}dw_3 + \sqrt{k_4x_4x_7}dw_4 \\
dx_7 &= (k_3x_2x_6 - k_4x_4x_7)dt + \sqrt{k_3x_2x_6}dw_3 - \sqrt{k_4x_4x_7}dw_4
\end{aligned}$$

The single slow reaction $SDH \rightarrow Z$ describes the conversion of the enzyme (SDH) into its inactive form at a rate of k_7x_5. When the reaction occurs, the number of molecules of x_5 is decreased by one and the concentration is decreased by $d_1 = 10^{-21}$ μM. The SHS model can be seen in Figure 12. The reset on the transition ($x_5- = d_1$) describes the effect of the single slow reaction on the concentration of x_5. For the SCD system, the classifications

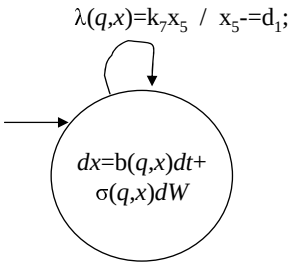


Figure 12: SHS model of SCD1

of the reactions do not change dynamically because the kinetic coefficients are significantly different and the chemical concentrations do not fluctuate widely.

SHS SCD Model with Medication Control (SCD2)

Drugs can help patients who are at high risk of developing sugar cataracts. These drugs work by inhibiting the enzyme SDH, thereby reducing the rate at which SDH reacts with other molecules in the system. This initially results in less sorbitol production; however, since the reversible reactions are tightly coupled, the results can have side effects such as increasing the fructose levels.

We create a new SHS model (SCD2), shown in Figure 13, of drug-modulated SCD to include the effect that the drug has on the system. The application of the drug is represented as a new discrete mode that represents drug-influenced dynamics where the reaction rates k_1,k_6, and k_7 are reduced by 50% to model the inhibition of the enzyme. Since the drug is metabolized slowly and the amount that the rates are reduced is directly proportional to the concentration of the drug, modeling a constant concentration is a reasonable approximation.

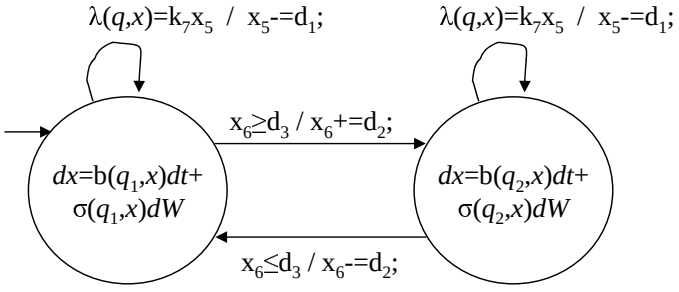


Figure 13: SHS model of medication-controlled SCD2

We model the drug administration based on an elevated level of fructose. It is assumed

that patients self-monitor and self-administer the medication. When the amount of fructose in the blood rises above a threshold $d_3 = 250$ μM, we use a guarded transition to drive the system to a new state that introduces the effect of the drug. When the fructose level drops back below d_3, we use another guarded transition to transition to the original state essentially removing the effect of the drug. We also include resets on the mode transitions to avoid infinitely fast switching that arises due to the stochastic nature of the Wiener process. The reset increases or decreases the fructose concentration by $d_2 = 1$ μM.

Fast switching could also be avoided by using guards that do not overlap (a slightly larger value for increasing guards and a slightly smaller value for decreasing guards). We selected to use state resets in order to reduce the number of the states required by the discrete approximation of the SHS. Although this type of guard eliminates the need for the reset on the transition, we found that this is not appropriate for this model, and it adds complexity to the analysis of the system.

SHS SCD Model with Probabilistically-Delayed Medication Effect (SCD3)

The SCD2 model is effective for demonstrating the effect of medication on the reactions; however, realistically the effect of the drug is not immediate because of variable drug metabolism rates. Drugs are generally administered in a form called a prodrug that allows the transport of the actual drug to the appropriate cells. This prodrug is metabolized into an active form of the drug at different rates for different people. Furthermore, once a patient discontinues taking a drug, the body can metabolize the residual drug at variable rates depending on many factors.

We develop a model (SCD3), seen in Figure 14, that incorporates two new states to model the delay of the conversion from prodrug to drug (q_2) and metabolism after dosage is discontinued (q_4). We use guarded transitions to model exiting the medicated and non-medicated states and entering the respective delay states. We then use probabilistic transitions to model the exit from the delay states to model the stochastic nature of the conversion and metabolism rates. The value $d_4 = 0.05$ is the rate of an exponential distribution that models the delay incurred by the conversion of prodrug to drug, and $d_5 = 0.05$ is the value that models the exponential distribution corresponding to the drug metabolism delay. These

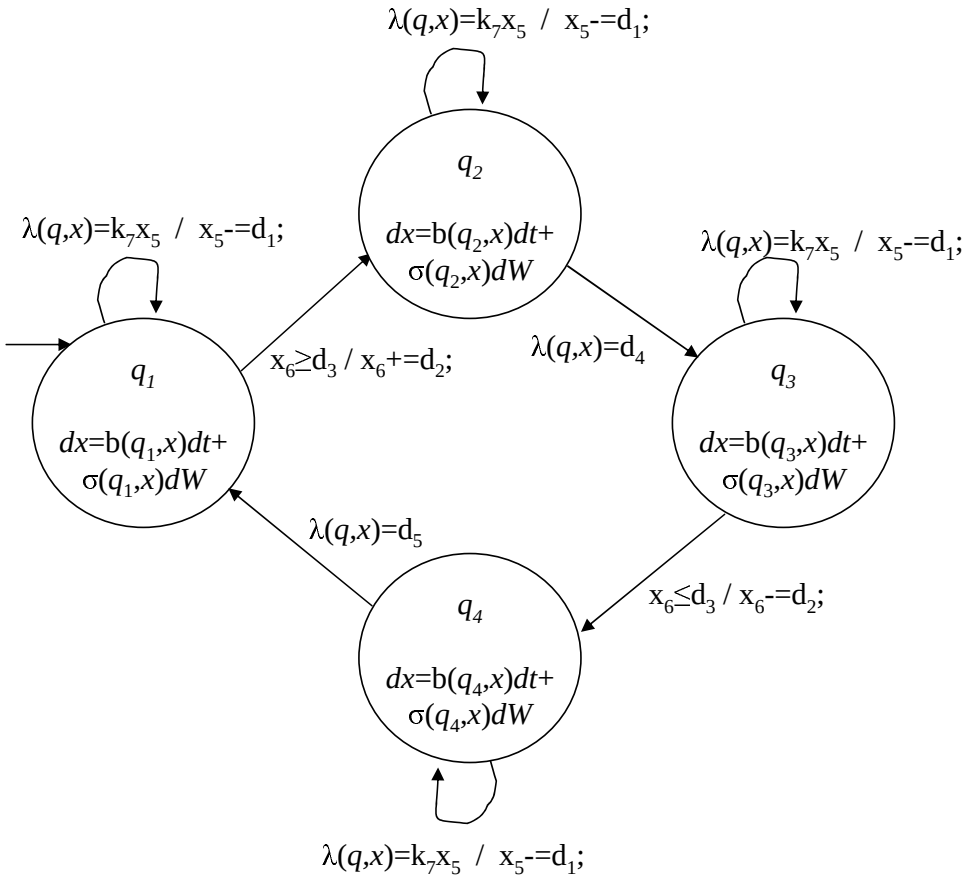


Figure 14: SHS model of medication-controlled SCD3 with delays

values were chosen so the average delay is on the order of one hour that is reasonable for the SCD system, but the values could be easily changed to model other types of medications. SHS can also incorporate the continuous state into the transition rate if necessary.

The continuous dynamics of the medicated (q_3) and non-medicated (q_1) states are consistent with SCD2. The dynamics of the delay state q_2 are the same as those in state q_1 to reflect the lack of change while the prodrug is being converted into the drug. The dynamics in the delay state q_4 model the metabolism of the drug after the administration is removed, so the kinetic coefficients are adjusted to reflect their half-life values. These coefficients can be adjusted to model various drugs.

A Biodiesel Processor

This section describes our SHS model of a biodiesel production system. We present this model to demonstrate our temperature modeling methodology, and we demonstrate its

correctness by validating the model in the next chapter. We also present the analysis of the biodiesel model using exhaustive verification in Chapter V. We compare the exhaustive verification analysis with verification using Monte Carlo methods in Chapter VI to demonstrate the correctness of both methods.

Biodiesel is made from vegetable oil and other chemicals by a process called transesterification [108]. The process involves six chemical species (Table 5) and six highly-coupled reactions (Table 6). Vegetable oil in its purest form is made up of triglycerides (TG); however, it breaks down into diglycerides (DG) and monoglycerides (MG) as it is heated. An alcohol, methanol (M), is combined with the TGs, DGs, and MGs to convert them into biodiesel esters (E) and glycerol (GL).

The concentrations of the chemical species for this process are given in Table 5. We choose chemical concentration ranges that are realistic for a 5 liter, experimental batch processor [139]. The reactions involved in the biodiesel process along with their kinetic rate equations are given in Table 6. The kinetic rate equations are used to calculate the kinetic coefficients of each reaction at various temperatures. Since temperature significantly effects the rates at which reactions occur, it is important to use accurate models of the kinetic coefficients. Our kinetic rate equations were derived using the Arrhenius equation and known dynamics of the reactions [108].

Table 5: Continuous state variables for the chemical concentrations of the reactions

Reactant	Variable	[Min, Max] (Moles)
TG	x_1	[0, 3]
DG	x_2	[0, 3]
MG	x_3	[0, 3]
E	x_4	[0, 9]
M	x_5	[0, 9]
GL	x_6	[0, 1]
T	x_7	[20,70]

It is critical to determine whether or not a biodiesel processor can produce high quality biodiesel that passes the American Society for Testing and Materials (ASTM) biofuels tests. Studies show that the amounts of GLs and TGs that are less than one percent still allow the resulting fuel to meet ASTM specifications [41]. The ASTM requirements also limit the

Table 6: Biodiesel reactions and kinetic rate equations

Reaction	Kinetic Rate
$TG + M \rightarrow DG + E$	$k_1 = 3.13 \times 10^7 e^{\frac{-6500}{T}}$
$DG + E \rightarrow TG + M$	$k_2 = 4.62 \times 10^5 e^{\frac{-4500}{T}}$
$DG + M \rightarrow MG + E$	$k_3 = 4.71 \times 10^{13} e^{\frac{-10100}{T}}$
$MG + E \rightarrow DG + M$	$k_4 = 7.89 \times 10^9 e^{\frac{-6500}{T}}$
$MG + M \rightarrow GL + E$	$k_5 = 4280 e^{\frac{-3200}{T}}$
$GL + E \rightarrow MG + M$	$k_6 = 17200 e^{\frac{-4600}{T}}$

amount of methanol that is dissolved in the biodiesel; however, to meet this requirement most biodiesel production systems use post-processing washing techniques that clean the excess methanol from the biodiesel after the main reactions fully complete [139].

To accurately model the reactions, the rate at which the individual reactions fire must be calculated using accurate temperature and pressure dynamics. The rate a_i at which chemical reactions occur can be calculated using the stoichiometry defined by the type of reaction. We consider reactions of the form: $V + X \rightarrow Y + Z$ where chemical species V, X, Y, and Z have concentrations v, x, y, and z, and k_i is the kinetic coefficient for reaction i. For these types of reactions the reaction rate is $a_i = k_i v x$. For other types of reactions, the rate can be calculated similarly [62].

SHS Biodiesel Process Model

We present our Variable Temperature BioDiesel (VTBD) model. The continuous dynamics in each state model the fluctuations in chemical concentrations and temperature. As seen in Figure 15, the model has two discrete states. One models the system heating q_1, and the other models the cooling state q_2. The glycerol separation is modeled using the self-loop transitions in each discrete state.

Since chemical dynamics are inherently stochastic, SDEs are an ideal modeling paradigm for the chemical concentrations. We use the method presented in [51] to model the concentrations of the chemicals in the system at each mode. The following equations model the biodiesel chemical species concentration fluctuations where x_i is the concentration of the ith chemical species and W_i is an element of an M_f-dimensional Wiener process.

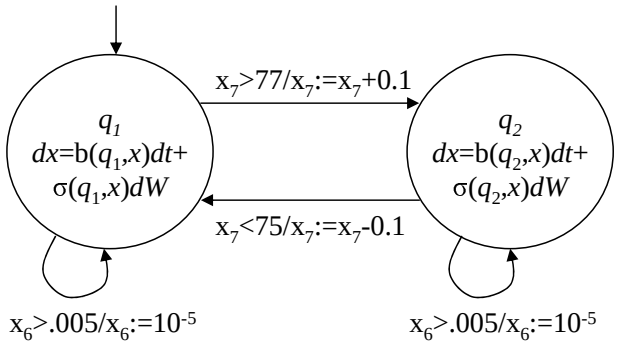


Figure 15: SHS model of the VTBD system

$$
\begin{aligned}
dx_1 &= (-k_1(x_7)x_1x_5 + k_2(x_7)x_2x_4)dt \\
&\quad -\sqrt{k_1(x_7)x_1x_5}dW_1 + \sqrt{k_2(x_7)x_2x_4}dW_2 \\
dx_2 &= (k_1(x_7)x_1x_5 - k_2(x_7)x_2x_4 - k_3(x_7)x_2x_5 + k_4(x_7)x_3x_4)dt \\
&\quad +\sqrt{k_1(x_7)x_1x_5}dW_1 - \sqrt{k_2(x_7)x_2x_4}dW_2 \\
&\quad -\sqrt{k_3(x_7)x_2x_5}dW_3 + \sqrt{k_4(x_7)x_3x_4}dW_4 \\
dx_3 &= (k_3(x_7)x_2x_5 - k_4(x_7)x_3x_4 - k_5(x_7)x_3x_5 + k_6(x_7)x_6x_4)dt \\
&\quad +\sqrt{k_3(x_7)x_2x_5}dW_3 - \sqrt{k_4(x_7)x_3x_4}dW_4 \\
&\quad -\sqrt{k_5(x_7)x_3x_5}dW_5 + \sqrt{k_6(x_7)x_6x_4}dW_6 \\
dx_4 &= (k_1(x_7)x_1x_5 - k_2(x_7)x_2x_4 + k_3(x_7)x_2x_5 \\
&\quad +k_4(x_7)x_3x_4 + k_5(x_7)x_3x_5 - k_6(x_7)x_6x_4)dt \\
&\quad +\sqrt{k_1(x_7)x_1x_5}dW_1 - \sqrt{k_2(x_7)x_2x_4}dW_2 + \sqrt{k_3(x_7)x_2x_5}dW_3 \\
&\quad +\sqrt{k_4(x_7)x_3x_4}dW_4 + \sqrt{k_5(x_7)x_3x_5}dW_5 - \sqrt{k_6(x_7)x_6x_4}dW_6 \\
dx_5 &= (-k_1(x_7)x_1x_5 + k_2(x_7)x_2x_4 - k_3(x_7)x_2x_5 \\
&\quad -k_4(x_7)x_3x_4 - k_5(x_7)x_3x_5 + k_6(x_7)x_6x_4)dt \\
&\quad -\sqrt{k_1(x_7)x_1x_5}dW_1 + \sqrt{k_2(x_7)x_2x_4}dW_2 - \sqrt{k_3(x_7)x_2x_5}dW_3 \\
&\quad -\sqrt{k_4(x_7)x_3x_4}dW_4 - \sqrt{k_5(x_7)x_3x_5}dW_5 + \sqrt{k_6(x_7)x_6x_4}dW_6 \\
dx_6 &= (k_5(x_7)x_3x_5 - k_6(x_7)x_6x_4)dt \\
&\quad +\sqrt{k_5(x_7)x_3x_5}dW_5 - \sqrt{k_6(x_7)x_6x_4}dW_6
\end{aligned}
$$

Biodiesel is made in processors that use heaters and thermostats to regulate the temperature because the chemical reactions involved are highly sensitive to temperature. Heating the reacting liquid is necessary to ensure production of high quality biodiesel, but using too much heat wastes time and money. Therefore, processors generally have built-in thermostats that control the temperature. To model this, we use two discrete states of the system, one for heating and one for cooling. We model the thermostat controller using guarded transitions between the heating and cooling states.

The SDE for modeling the temperature in each state is given by

$$dx_7 = \begin{cases} .02(300 - x_7)dt + .01dW & q_1 \\ .01(-x_7)dt + .01dW & q_2 \end{cases} \tag{5}$$

We designed the equations to model traditional heating and cooling dynamics and chose the constants to model realistic heating and cooling characteristics of similar liquids and heaters. We assume that the chemical reactions are neither exothermic nor endothermic, which is realistic for this system. The temperature (x_7) affects the reaction rates, so accurate modeling is imperative. The resets ($R((q, x), dy)$) are used to ensure that the dynamics are sufficiently moved to avoid infinite switching behavior.

As the chemical reactions produce biodiesel, glycerol (GL) is formed as a byproduct of the reaction. Since the presence of glycerol inhibits the successful production of high quality biodiesel, separation of the glycerol from the reacting liquid is necessary. Glycerol is significantly denser than biodiesel so it can be removed using gravity settling or a centrifuge depending on the type of processor [139]. We model the glycerol separation using self-transitions in the heating and cooling states. Once the concentration of glycerol rises above a certain level, the transition is enabled and the reset on the transition reduces the concentration of the glycerol.

Glycolysis

We present the glycolysis model to demonstrate our modeling methodology for a large, complex biochemical system. We use this model to present our simulation methods as well

as the Monte Carlo methods since it is too large for exhaustive verification. Further, the glycolysis model is used to present variance reduction methods to demonstrate accuracy and efficiency gains.

Glycolysis is a series of biochemical reactions that converts carbohydrates into various waste products and energy in a currency useful to cells (ATP). As it is a fundamental process to all living cells, it has been studied and modeled extensively in many organisms. Baker's yeast (Saccharomyces cerevisiae) is a well-established model organism for the study of glycolysis [70]. Although the individual steps of glycolysis have been thoroughly examined, the interaction of glycolytic enzymes, substrates, and products with the intracellular environment is not fully understood. Modeling and simulating glycolysis using SHS can further our understanding of contextual cellular respiration.

SHS Glycolysis Model

Twenty-two chemical species have been identified that play an important role in glycolysis and can be seen in Table 7. The chemical species interact in a series of 37 interconnected chemical reactions (as seen in Table 8). Glucose (Glc) is added to the system, and glycogen ($Glyc$), ethanol ($EtOH$), ATP, and other minor chemicals are produced. The reaction rates for the system are developed in previous work [70]. The network of chemical reactions can be seen in Figure 16.

Table 7: Glycolysis chemical species

Chemical	Starting value	Chemical	Starting Value	Chemical	Starting Value
Glc_x	4.25	$NADH$	0.3	AMP	0.25
Glc	2.75	PEP	0.041	BPG	0.0003
ATP	2.1	Pyr	22	CN_x^-	5.6
$G6P$	4.4	ACA	1.6	NAD^+	0.7
ADP	1.5	ACA_x	1.4	$Glyc_x$	2.8
$F6P$	0.6	$EtOH$	20	$DHAP$	3.1
FBP	5	$EtOH_x$	17		
GAP	0.1	$Glyc$	4.4		

The model presented in [70] is a deterministic model, but the chemical reactions in the real system actually behave in a stochastic manner due to the uncertainty of molecular

Table 8: Glycolysis chemical reactions

Reaction	Reaction
$Glc_x \rightleftharpoons Glc$	$ATP + AMP \rightleftharpoons 2ADP$
$Glc + ATP \rightarrow G6P + ADP$	$EtOH_x \rightarrow$
$G6P \rightleftharpoons F6P$	$Glyc_x \rightarrow$
$F6P + ATP \rightleftharpoons FBP + ADP$	$ACA_x \rightarrow$
$FBP \rightleftharpoons GAP + DHAP$	$ACA_x + CN_x^- \rightarrow$
$DHAP \rightleftharpoons GAP$	$CN_x^- \rightleftharpoons$
$GAP + NAD^+ \rightleftharpoons BPG + NADH$	$Glc_x \rightleftharpoons$
$BPG + ADP \rightleftharpoons PEP + ATP$	$Glyc \rightleftharpoons Glyc_x$
$PEP + ADP \rightarrow Pyr + ATP$	$ACA \rightleftharpoons ACA_x$
$Pyr \rightarrow ACA$	$G6P + ATP \rightarrow ADP$
$ACA + NADH \rightarrow EtOH + NAD^+$	$ATP \rightarrow ADP$
$EtOH \rightleftharpoons EtOH_x$	$DHAP + NADH \rightarrow Glyc + NAD^+$

motion. Therefore, we incorporate stochastic dynamics into the model using the method described previously. We also incorporate discrete dynamics into the glycolysis model to capture the concept of 'feeding' the yeast. Glucose must be added to the system to continue production of the energy molecules, and when the concentration of glucose diminishes, the amount of energy molecules that the system can produce decreases. In many organisms, this reduction in energy output triggers mechanisms that encourage the introduction of more glucose (i.e, feeding).

We model the glycolysis system using a SHS with two states: saturated and deficient as shown in Figure 17. In the saturated state the glucose intake rate is less than the in deficient state. Switching between the states is regulated by the concentration of ATP (x_3). When the amount of ATP drops below a certain level, the discrete state switches from the saturated to the deficient state. If the level of ATP climbs back above a certain level, then the state switches back to the saturated mode from the deficient mode.

Water/Electrolyte Balance

We present a model of the water/electrolyte balance system in humans. Our model is a relatively simple, extensible model intended to demonstrate the ability of SHS to model macro-sized biochemical systems. It is used in Chapter *IV* to demonstrate the accuracy and efficiency of our proposed SHS simulation algorithms.

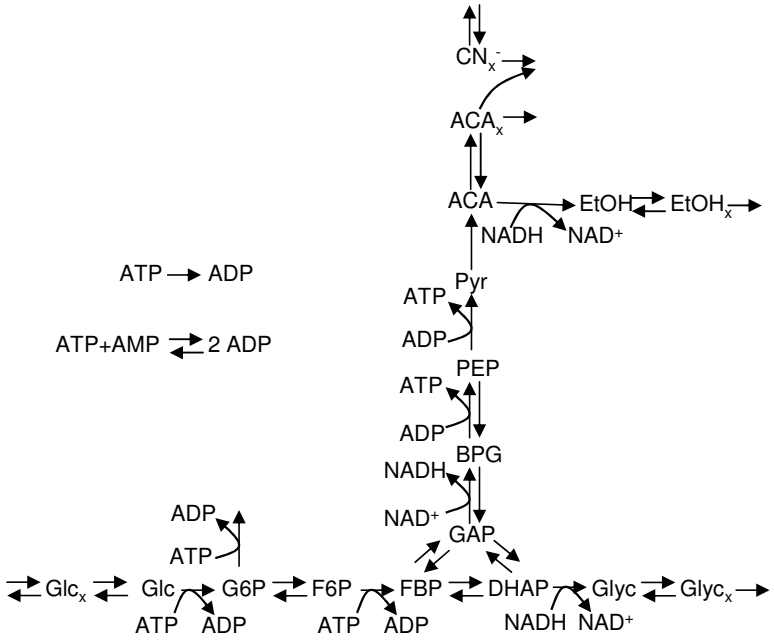


Figure 16: Network of glycolysis reactions

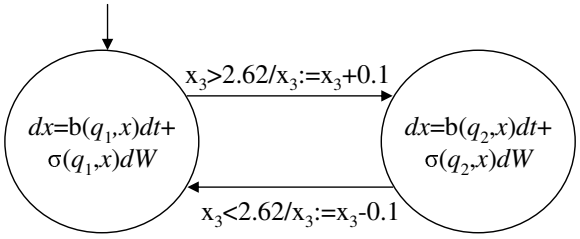


Figure 17: SHS model of glycolysis

Water/electrolyte balance regulation in mammals is vital to life. If too much salt is present, dehydration occurs, leading to discomfort, performance degradation, and even death. If too much water is present, arterial pressure rises dangerously and the nervous system begins to malfunction. Therefore, virtually every living organism has a system that regulates water balance.

Anti-Diuretic Hormone (ADH) is a nine amino acid peptide hormone secreted by the hypothalamus. ADH is released when the body senses the intake of too much salt or a shortage of water. Upon these conditions ADH signals to the kidneys to retain water to compensate and bring the body back to equilibrium. Upon secretion, ADH travels through the bloodstream to exert the majority of its effects on specific receptors (arginine vasopressin receptor

2; AVPR2) in specialized cells within the kidney tubules. When ADH binds AVPR2, a chain of intracellular signaling events takes place. The succession of signaling events ultimately results in additional insertion of extra water channels (aquaporins;AQP2) into the apical membrane of the cell. Aquaporins allow water to pass out of the nephrons and be re-collected into the cells. Once the water is reabsorbed, a smaller, more concentrated amount of urine is excreted [123].

The insertion of AQP2 channels into the cell's outer membrane is a highly regulated, multistep process. AQP2 is synthesized in the cell and inserted into intracellular membrane structures called vesicles. When called upon by ADH-AVPR2 interaction and resulting intracellular signaling, attachment and tethering proteins specifically direct the vesicles to fuse with the outer membrane of the cell. The fusion event results in the addition of the AQP2 molecules to the outer membrane. The total number of available AQP2-containing vesicles and the attachment and tethering proteins are both inherently limited in any given cell resulting in a saturation point for sensitivity of the cell to ADH [107].

When ADH is withdrawn, AQP2 accumulates in special membrane domains (clathrin-coated pits), which are subsequently engulfed (endocytosed) by the cell. Endocytosed AQP2 receptors are then recycled within the cells, ready for the next ADH signal. AQP2 is continuously and quickly recycled between the cell surface and intracellular compartments, rebounding between upper and lower limits for AQP2 cell surface localization. This behavior results in a reflection of the observed effects at the ADH saturation limit [107].

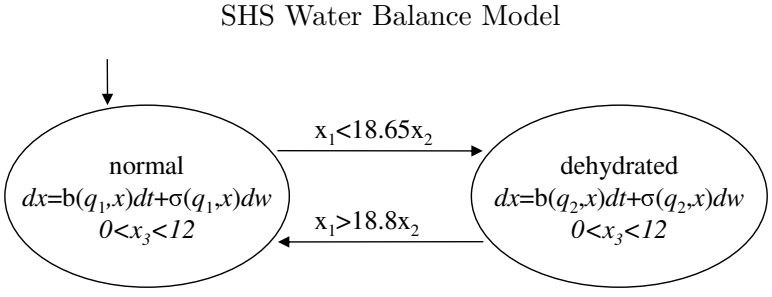


Figure 18: Water balance SHS model

We develop a SHS model of the water/electrolyte balance system, seen in Figure 18. The

SHS has been adapted from the SDE model in [92] to include the thirst/dehydration mechanism described in [77]. The model includes two discrete states: normal and dehydrated. Transitions between the normal and dehydrated modes are defined by the transition guards in Figure 18 and are based on the ratio of water to salt (or electrolyte concentration) in the body derived from data in [77].

We define three continuous states: total body water x_1, total body salt x_2, and ADH x_3 within each discrete state. The dynamics for the water and salt variables were based on simple input/output differences in the system with an added diffusion term that models uncertainty and system variability [92]. SDEs are used with constant diffusion because of the uncertainty of molecular interactions in these types of biochemical systems. Fluid output is directly dependent on the ADH concentration that is in turn affected by the fluid/salt ratio in the body.

The following SDEs describe the continuous dynamics in the normal state

$$\begin{aligned} dx_1 &= (f_{in} - 45x_3^{-.76})dt + .1dw_1 \\ dx_2 &= (s_{in} - s_{out})dt + .1dw_2 \\ dx_3 &= (-4.5)dt + .1dw_3 \end{aligned}$$

where f_{in} describes the amount of fluid input to the system per unit time, s_{in} describes the amount of salt input to the system per unit time, s_{out} describes the amount of salt output from the system per unit time, and $w = [w_1, w_2, w_3]^T$ is a three-dimensional Wiener process.

The next set of equations describe the dynamics when the body is in the dehydrated state determined by the electrolyte concentration.

$$
\begin{aligned}
dx_1 &= (f_{in} - 45x_3^{-.76})dt + .1dw_1 \\
dx_2 &= (s_{in} - s_{out})dt + .1dw_2 \\
dx_3 &= \left(\frac{80 * x_2}{x_1}\right)dt + .1dw_3
\end{aligned}
$$

The constants for the continuous dynamics were adapted from [92] to match the experimental data in [77]. We fit the experimental data to curves and determined appropriate adaptations for the dynamics when necessary. The values we used for our experiments can be found in Table 9. The fluid input f_{in} can be modeled as a continuous stream or discrete input, so for simplicity we consider only the continuous stream. Our focus is primarily on the water balance, so we modeled s_{in} and s_{out} as constant functions; however, these could be easily extended to model more realistic behavior if salt balance is the focus of the analysis.

Table 9: Water balance model coefficients

Variable	Value
f_{in}	40
s_{in}	2
s_{out}	2

Because ADH is only valid for non-negative values, a reflective boundary is defined for x_3 at the value of zero. We also define a reflective boundary at $x_3 = 12$ to mimic the saturation limit of ADH in the kidneys. The limit is defined by the invariants in the system $x_3 \in [0, 12]$. This range may not be the same for every person, but seems reasonable based on experimental data from [77].

Summary

In this chapter we present a biochemical system modeling methodology using SHS as well as several models that demonstrate the capabilities and accuracy of the modeling method. The models are intended to provide realistic examples to be used with analysis techniques.

We present three models of SCD that demonstrate multiple medication administration modeling methods. We also present a biodiesel process model that incorporates temperature-dependent chemical dynamics and a heating controller. We present a model of glycolysis that models the conversion of sugars into usable energy molecules for a cell. The final model we present is the water and electrolyte balance system in humans. This model demonstrates the ability of SHS to capture macro behavior in a realistic system.

The modeling methodology and models we present provide a sample of the types of systems that can be accurately and efficiently modeled with SHS. The models provide a platform for future development as well as a basis for comparison for SHS analysis methods.

CHAPTER IV

SIMULATION OF SHS

Biochemical processes are inherently stochastic and often contain both continuous and discrete behavior, so Stochastic Hybrid Systems (SHS) provide an ideal framework for modeling these types of systems [29, 75]. Simulation methods for SHS can be implemented by combining simulation methods for continuous and discrete dynamics. However, SHS simulation methods are challenging because the complex dynamics can introduce error and inefficiencies. Adaptive time stepping methods for Stochastic Differential Equations (SDEs) have been proposed [90], and hold promise to significantly improve the efficiency and control the error of SHS simulation methods.

Simulation of SHS using fixed-step methods is challenging because of the inherent inaccuracies of simulating stochastic dynamics and the small step sizes required to achieve good approximations [123]. Stochastic continuous dynamics can be difficult to accurately and efficiently approximate, and while simulation methods for SDEs exist, the implementation of the methods for high-dimensional systems is non-trivial [80]. Discrete transitions can also be difficult to accurately and efficiently approximate because of the inherent challenge of detecting boundary crossings [123]. SHS simulation algorithms require the use of significantly small step sizes to improve accuracy, but this comes at the cost of efficiency.

Adaptive time step methods for SHS hold promise to improve accuracy and efficiency, but they are challenging because error can be introduced in many forms and exact error computation is not possible for complex systems. Error approximation methods exist, but they must be used carefully to ensure that they approximate relevant error [90]. Further, using various step sizes can cause difficulty when approximating the Brownian motion of the system and can introduce bias if not handled carefully. Adaptive time stepping for SHS is further complicated by the presence of boundaries that must be handled appropriately to avoid introducing unnecessary error.

In this chapter we describe simulation methods for SHS. We describe two methods for simulating SDEs as well as traditional and advanced methods for detecting absorbing

and reflecting boundary crossings. We develop two fixed step SHS simulation algorithms using the traditional and advanced simulation methods. We also develop an adaptive time stepping SHS algorithm based on an adaptive SDE time stepping algorithm. We describe several methods for approximating the error, choosing appropriate time step adjustments, and generating noise for variable SHS time steps.

We present experimental results for several systems to demonstrate our methods. We compare our SHS simulation method results with simulations of the SSA for the SCD system to demonstrate the correctness of our modeling and simulation methods [124]. We also compare our SHS simulation method with the results of experimental data for the biodiesel model to show the accuracy of both the simulation method and the model [126].

In this chapter we also demonstrate the improvements of our advanced SHS simulation algorithms using the water balance model [123]. We present results of step size comparisons and a demonstration of the adaptive time stepping algorithm. We also present an analysis of the accuracy and efficiency of the adaptive time stepping SHS algorithm using the Variable Temperature BioDiesel (VTBD) model.

Fixed Time Step Simulation

Simulation of SHS requires the combination of simulation methods for SDEs, detection of switching boundaries, approximation of reflecting boundaries, and detection of probabilistic transitions. At each time step, the values of the continuous variables must be updated, boundaries must be tested for crossings, and probabilistic transitions must be tested for firings. We present the individual simulation methods and we describe the combination of the methods to create SHS simulation algorithms.

Numerical Integration of SDEs

Simulation of SDEs can be performed using Taylor schemes of various orders. The simplest Taylor approximation scheme is the Euler Maruyama (EM) method that is a first-

order approximation. The kth component of the EM scheme is given by

$$X_{n+1}^k = X_n^k + b^k \Delta t + \sum_{j=1}^{m} \sigma^{k,j} \Delta W^j$$

for $k = 1, 2, ..., d$ where ΔW^j is the normally-distributed increment of the jth component of the d-dimensional Wiener process W assuming a d-dimensional drift coefficient b and a $d \times m$ diffusion coefficient σ.

To formalize the notion of accuracy, we present the concept of order of convergence. The order of convergence is used to quantify the quality of the approximation when considering simulation of stochastic systems. An approximation $X^{\Delta t}(T)$ at time T with step size Δt converges with order γ strongly to the actual trajectory $x(T)$ if there exists $c > 0$ such that $E\left(\left|x(T) - X^{\Delta t}(T)\right|\right) \leq c\Delta t^{\gamma}$. $X^{\Delta t}(T)$ converges with order γ weakly to $x(T)$ if there exists $c > 0$ such that $E\left(\left|f(x(T)) - f(X^{\Delta t}(T))\right|\right) \leq c\Delta t^{\gamma}$ for a given class of measurable functions f [80]. Strong convergence implies that the trajectory is a possible trajectory of the system, and weak convergence implies that the computed trajectory only preserves the moments of the actual trajectory. The EM method is simple to implement, but achieves a strong convergence of $\gamma = 0.5$ and weak convergence $\gamma = 1.0$, so small time steps must be used to generate accurate approximations.

The Milstein Method (MM) is a second-order Taylor scheme. The higher order terms require more computation; however, the approximation maintains an acceptable efficiency for most systems.

The kth component of the MM scheme is described by

$$X_{n+1}^k = X_n^k + b^k \Delta t + \sum_{j=1}^{m} \sigma^{k,j} \Delta W^j + \sum_{j_1,j_2=1}^{m} L^{j_1} \sigma^{k,j_2} I_{(j_1,j_2)}$$

where

$$L^j = \sum_{k=1}^{d} \sigma^{k,j} \frac{d}{dx^k} \text{ and } I_{(j_1,j_2)} = \int_{\tau_n}^{\tau_{n+1}} \int_{\tau_n}^{s_1} dw_{s_2}^{j_1} dw_s^{j_2}.$$

For the cases where $j_1 = j_2$, the multiple stochastic (Stratonovich) integral can be calculated

by

$$I_{(j_1,j_1)} = \frac{1}{2}((\Delta W^{j_1})^2 - \Delta t)$$

$I_{(j_1,j_2)}$ cannot, in general, be calculated using only ΔW^j values. To approximate $I_{(j_1,j_2)}$, multiple stochastic integrals are used in the following equation assuming $j_1 \neq j_2$.

$$\begin{aligned} I^p_{(j_1,j_2)} &= \Delta t(\frac{1}{2}\xi_{j_1}\xi_{j_2} + \sqrt{p_p}(\mu_{j_1,p}\xi_{j_2} - \mu_{j_2,p}\xi_{j_1})) + \\ &\quad \frac{\Delta t}{2\pi}\sum_{r=1}^{p}\frac{1}{r}(\zeta_{j_1,r}(\sqrt{2}\xi_{j_2} + \eta_{j_2,r}) - \zeta_{j_2,r}(\sqrt{2}\xi_{j_1} + \eta_{j_1,r})) \end{aligned}$$

where

$$p_p = \frac{1}{12} - \frac{1}{2\pi^2}\sum_{r=1}^{p}\frac{1}{r^2}$$

$$\xi_j = \frac{1}{\sqrt{\Delta t}}\Delta W^j$$

and $\mu_{j,p}$, $\eta_{j,r}$, and $\zeta_{j,r}$ are independent Gaussian random variables with mean 0 and standard deviation 1 for $j = 1, ..., m$ and $r = 1, ..., p$. The accuracy of the approximation $I^p_{(j_1,j_2)}$ of $I_{(j_1,j_2)}$ can be improved by using larger values of p. To obtain a strong convergence of order $\gamma = 1.0$, $p = p(\Delta t) \geq \frac{K}{\Delta t}$ must be chosen where K is some positive constant [80].

Taylor schemes for solving SDEs can have strong order of convergence of $\gamma = 0.5$ to $\gamma = 3.0$ and weak order of convergence of $\gamma = 1.0$ to $\gamma = 6.0$ depending on the number of approximating terms [80]. The computation of higher order terms requires many more operations and can be prohibitively complicated and expensive; therefore, a trade off must be reached to achieve the appropriate accuracy and efficiency. Using MM is effective for improving accuracy while maintaining sufficient efficiency.

Absorbing Boundaries

During the execution of a SHS, the process can hit a switching boundary defined by the invariants. At a switching boundary the continuous process is halted and re-started in a new state after executing any transition resets. Because the process is stopped when a boundary is encountered, switching boundaries can be treated as absorbing boundaries that are restarted in a new state after being absorbed in the current state [56]. It is important

to accurately estimate the time and location that the process is absorbed to minimize the error introduced into the approximation.

The easiest way to detect an absorbing boundary is to check the state against the invariants at each step of the approximation. Let us assume the state at time t is $X(t)$. If $X(t) \in X^q$, but $X(t + \Delta t) \notin X^q$, then the process is rolled back to time t and re-started in the new state. This method has a strong order of convergence of $\gamma = 0.5$ [111].

An improved method for absorbing boundary crossing detection based on stochastic sampling was developed in [56]. The approach can be used with boundaries that are hyperplanes or sufficiently smooth. The biochemical models we consider have boundaries that are hyperplanes, so this approach is valid for these systems. The probability that the state trajectory has hit the boundary between t and $t + \Delta t$ is

$$P(hit) = exp\left(\frac{-2(n.(X_t - X_{ab}))(n.(X_{t+\Delta t} - X_{ab}))}{n.(\sigma(X_t)\sigma^*(X_t)n)\Delta t}\right)$$

where the switching boundaries are hyperplanes $\partial X^q = \left\{x \in \mathbb{R}^{d(q)} : n.(x - X_{ab}) = 0\right\}$, n is the unit vector normal to the boundary ∂X^q, $X_{ab} \in \mathbb{R}^n$ is the position of the absorbing boundary, and X_t is the computed continuous state at time t. For simplifying the notation we choose to describe the diffusion $\sigma(X_t)$ as only depending on the continuous state, but the actual diffusion may depend on the discrete state as well. This improved method achieves a weak order of $\gamma = 1.0$ assuming that the boundary is sufficiently smooth [56].

Reflecting Boundaries

Invariants can define reflective boundaries in addition to switching boundaries. Reflective boundaries are those where the process is reflected obliquely when it encounters the boundary [56]. For example, all biochemical systems are limited to nonnegative concentrations of any chemicals, or some biochemical processes also have saturation limits that impose upper limits on concentrations. In both cases when the process reaches the boundary, it is reflected to mimic the behavior of the real system.

The traditional way to handle reflective boundaries is to detect the boundary crossing and reset the state to a position within the valid state space [56]. The most common way

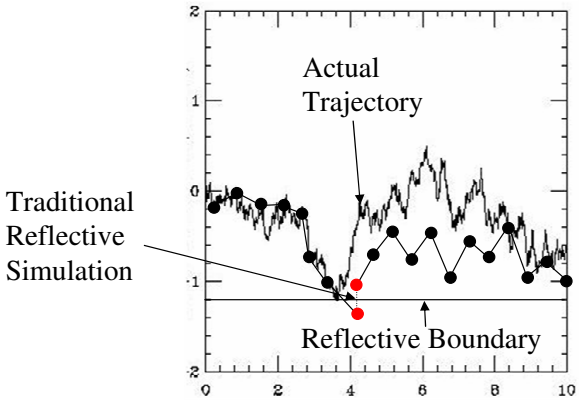


Figure 19: Boundary reflection problem

to reset the state is to place it inside the state space the same distance that it covered after it crossed the boundary. This type of reflection guarantees that the invariants are always satisfied, but it is not always accurate for real systems. For example, biochemical systems require non-negative chemical concentrations, so if the simulation is reflected using the traditional method, the number of atoms involved in the system may be reset to a value that invalidates conservation of matter properties. A simple example of this is shown in Figure 19.

We formally define the boundary reflection problem. Let us assume a system has an invariant X^q with a reflective boundary, and the state at time t is $X(t)$. If $X(t) \in X^q$, but $X(t + \Delta t)$ is computed to be outside of X^q, then the process is reflected in the direction normal to the boundary $X(t + \Delta t).n = X(t).n$ where n is the unit vector normal to the boundary. The traditional method of boundary reflection has a weak order of convergence of $\gamma = 0.5$ [56].

The improved method described in [56] defines a new diffusion process that adds the effect of the reflection to the original SDE:

$$dx = b(q, x)dt + \sigma(q, x)dw + n(q, x)dk$$

where $n(q, x)$ is a unit vector normal the boundary at state (q, x), $k = \int_0^t 1_{X \in \partial D} dk$, ∂D is the reflective boundary, and X_{rb} is the position of the reflective boundary.

The approximation of the process is calculated using:

$$X_{t+\Delta t} = X_t + b\Delta t + \sigma\Delta W + n\Delta k$$

where ΔW is a normally-distributed pseudo-random number and $\Delta k = k_t - k_{t+\Delta t}$. Approximating k_t is achieved using the technique described in [56]:

$$\begin{aligned} k_t &= \max(0, z_t).n \\ z_t &= X_t - X_{rb} + \frac{1}{2}\left(\sigma W + bt + \sqrt{|\sigma|^2 V + (\sigma W + bt)^2}\right) \end{aligned}$$

where $V = \epsilon(1/2t)$ is an exponentially-distributed random variable independent of W. This equation is derived from the solution to the Skorohod problem and results in a weak order 1.0 approximation of the reflecting boundary [56].

Probabilistic Transitions

Firing of the probabilistic transitions (according to the transition rate λ) can be handled by the technique described in [18]. A graphical representation of this algorithm can be seen in Figure 20. First, a new process Z must be defined

$$Z(t) = -U + e^{-\int_t^{t_{lt}} \lambda(x(s))ds}$$

where $U \in [0, 1]$ is a uniformly-distributed random number and t_{lt} is the time of the last probabilistic transition. A sample is drawn from the uniform distribution and Z is tested at each time step. When Z crosses 0, the transition is fired.

SHS Simulation Algorithms

Our first, and most simple, algorithm is the Hybrid Euler-Maruyama *HEM* method. We use the Euler-Maruyama method for computing the SDE values, and we use the traditional (non-stochastic) methods for detecting boundaries. Discrete transitions are incorporated into the EM approximations by analyzing the state between steps of updating the continuous dynamics. If a guard condition on a transition is satisfied, then the transition is fired and

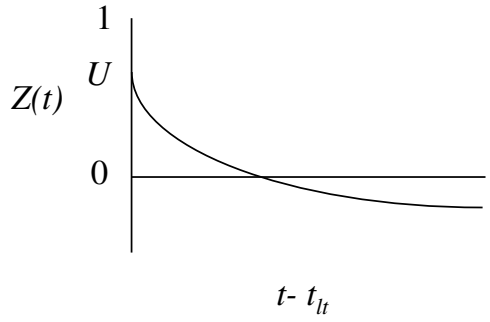


Figure 20: Probabilistic transition firing method

resets are executed. Once the state is updated, the EM algorithm continues in the new state. Probabilistic transition firing is determined for *HEM* using the technique described earlier. We draw a sample from a uniform distribution and test the exponential decay at various times to determine the jump time for each probabilistic transition. When the exponential decay is greater than or equal to the random value, the transition is fired. The pseudocode for a step of *HEM* is given below where Δt is the step size, *guard* is the boolean guard on a single discrete transition, X_{rb} is the location of the reflective boundary, and t_{tl} is the time the last probabilistic transition fired. Multiple guarded or probabilistic transitions may be included by adding multiple tests.

Algorithm 4.1: HEMSTEP(X_t)

$X_{t+\Delta t} = X_t + b\Delta t + \sigma\Delta W$
if $guard == true$
  then $FireGuardedTransition$
if $X_{t+\Delta t} < X_{rb}$
  then $ReflectBoundary$
if $U_1 = rand(0,1) < exp(-\lambda(t - t_{tl}))$
  then $FireProbabilisticTransition$
$t = t + \Delta t$
return (X_t)

Our second simulation algorithm *HMM* incorporates the MM, stochastic absorbing, and reflective boundary simulation methods. To include the improved reflective boundary

method we store the previous Δk value and calculate the new Δk at each step. If the trajectory is close to a reflecting boundary, we add Δk to the MM computation. Because there may be multiple absorbing boundaries that could be hit at the same time, at each time step we calculate the probability of hitting all nearby boundaries. We then select the boundary with the highest hitting probability and compare the probability to a uniformly-distributed number U_1. When $U_1 < max(P)$, then we consider the boundary to be hit, and we execute the transition resets and restart the process in the new state. The pseudocode for a step of *HMM* is as follows where $x_{ab,\xi}$ is the location of absorbing boundary ξ, n is the direction normal to the associated reflected or absorbing boundary, and V is an exponentially distributed random variable independent of U_1.

Algorithm 4.2: HMMSTEP(X_t)

$k_{t+\Delta t} = X_t - X_{rb} + \frac{1}{2}\left(\sigma W + b(t+\Delta t) + \sqrt{|\sigma|^2 V + (\sigma W + b(t+\Delta t))^2}\right)$

$\Delta k = \max(k_{t+\Delta t}, 0).n - prev\Delta k$

$X_{t+\Delta t} = X_t + b\Delta t + \sigma\Delta W + \sum_{j_1,j_2=1}^{m} L^{j_1}\sigma^{k,j_2} I_{(j_1,j_2)} + n(\Delta k)$

$GuardedProb = max(exp(\frac{-2(n.(X_t - X_{ab,\xi}))(n.(X_{t+\Delta t} - X_{ab,\xi}))}{n.(\sigma\sigma^*(X_t)n)\Delta t}), \xi)$

if $U_1 = rand(0,1) < GuardedProb$

  then $FireGuardedTransition$

if $U_2 = rand(0,1) < exp(-\lambda(t - t_{tl}))$

  then $FireProbabilisticTransition$

$t = t + \Delta t$

return (X_t)

Error is introduced into the calculated SHS trajectory in several different ways. Approximation of the SDE introduces higher-order errors that are not calculated due to computational inefficiency. Error due to the use of pseudo-random numbers is typically not a concern for smaller simulations, but large simulations must use pseudo-random generators that do not repeat as often as the efficient generators to avoid this type of error. Finally, step size inherently introduces error in the SDE and boundary calculations as described earlier by the order of convergence γ.

The approximations using the EM or MM method, boundary methods, and probabilistic

transitions converge to the actual solution individually as the step size is decreased to zero, so their combination also converges to the correct solution. By combining methods with higher order convergence, we reduce approximation error more quickly than the lower order methods thereby improving efficiency and accuracy. The traditional absorbing and reflecting boundary algorithms have a weak order of convergence of $\gamma = 0.5$, while the improved methods both have a weak order of convergence of $\gamma = 1.0$ [56]. Therefore the *HEM* algorithm has a strong order of convergence of $\gamma = 0.5$, and the *HMM* algorithm has a strong order of convergence of $\gamma = 1.0$.

Accurate simulation of the trajectory near intersections of boundaries is a difficult problem, and must be handled carefully to minimize error. When the trajectory is in close proximity to multiple reflecting or absorbing boundaries, our algorithm considers the boundary with the highest hitting probability at each time step.

Adaptive Time Step Simulation

Error in a simulation method due to the step size can be decreased by decreasing the step size, but this comes at the cost of efficiency. Dynamically adjusting the time step of the simulation has been shown to increase the accuracy and efficiency of the approximation by allowing the step size to adjust to compensate for variable step size error. However, adaptive time stepping for stochastic systems is difficult because of the challenge of error approximation in the presence of stochastic dynamics [90]. Adaptive time stepping for SHS is further complicated by the discrete discontinuities, so additional care must be taken when a simulation trajectory approaches a boundary.

Background

Fixed step integration methods are easy to implement and are effective for generating approximations to differential equations, but they can be unnecessarily inefficient. Adaptive time stepping can improve efficiency by adjusting the time step of the approximation dynamically based on the error of the approximation.

Exact error amounts cannot be determined for general systems, so error estimations must be used. Error estimation methods aim to determine the amount of error generated

in a time step by examining the dynamics of the simulation. If the estimated error is too large, then the given time step should be decreased. Conversely, if the error is sufficiently small, the step size can be increased because the error introduced will be relatively small.

Accurate approximations of the error due to the step size must be made to ensure the step size is adjusted appropriately. For ordinary differential equations, error approximations and step size adjustments are fairly strait forward [90]. However, adaptive time stepping for SDEs is not as simple for multiple reasons. Not only is error introduced by several sources (that all must be accurately estimated), but also the Brownian path must be computed accurately when the step size changes to ensure randomness is preserved. Therefore, we begin by examining error estimates for SDEs.

SDE Error Approximation

Time discretization error for SDEs can be categorized into two types: drift and diffusion error. Neither type of error can be computed exactly because there are no analytical methods for computing the error of SDEs. However, both types of error can be estimated separately and decisions about the time step can be made based on the amount of either or both forms of error.

The error introduced by the diffusion term can be estimated by computing higher-order approximation terms. Given a SDE such as Equation 3, we examine the higher-order terms of the Strantonovich-Taylor expansion of the original SDE $J_{10}b'\sigma$, $J_{01}\sigma'b$, $\frac{1}{6}J_1^3\sigma'\sigma\sigma$, and $\frac{1}{6}J_1^3\sigma'\sigma'\sigma$ where J_1,J_{10}, and J_{01} are multiple Strantonovich integrals [90]. Only the last term can be computed efficiently, so it is the best term to estimate the diffusion-influenced error:

$$E_\sigma = \Delta W^3 \sigma'^2 \sigma.$$

This method has been shown to be effective for estimating the diffusion error previously in [90].

Error can also be introduced by the drift term, so we must also consider this error in our estimation methods. ODE-like error computation methods can be used to estimate the drift error using the SDE. Using the $O(h^2)$ error terms from the Milstein error expansion,

the drift error can be estimated by

$$E_b = \Delta t^2 b' b.$$

This method has been shown to be effective for estimating the drift error previously in [90].

Adaptive Time Stepping for SDEs

Before each step of the approximation, the drift and diffusion errors of the approximation are computed (E_b,E_σ), and the step is rejected or accepted depending on the amount of either type of error. If the step is rejected because either E_b or E_σ is too large, the step size is reduced to decrease the error until both error estimates are sufficiently small. If the two types of error are both determined to be smaller than a threshold, the step size can be increased to improve efficiency. Step sizes are typically halved and doubled in stochastic systems to simplify the computation of the Wiener process [90].

Brownian motion must be appropriately approximated for the variable time steps to ensure bias is not introduced. To simplify the process, a binary tree structure is used to store the noise values. The Wiener process is sampled at fixed intervals $\Delta w_k = w(k) - w(k-1)$ for $k = 1..N$. Intermediate intervals on level j of the tree are calculated if needed by $\Delta w_{2k-1,j+1} = \frac{1}{2}\Delta w_{k,j} + y_{k,j}$, $\Delta w_{2k,j+1} = \frac{1}{2}\Delta w_{k,j} - y_{k,j}$ for $j = 1, 2, ..$ where $y_{k,j}$ is a normally distributed random variable with mean 0 and variance 2^{-j} [45].

Adaptive Time Stepping for SHS

The dynamics of SHS introduce further complication into the estimation of the error for an approximation method. Significant error can be introduced for near boundaries as we showed in the previous section, and larger step sizes exacerbate this error. Traditional error estimation methods for SDEs do not consider boundaries, so large step sizes near boundaries are possible. Therefore, we must test and prevent large step sizes near reflecting or absorbing boundaries before a step size is accepted.

Error introduced near boundaries can be significantly reduced by shrinking the step size when a trajectory is near a boundary. Therefore, we test if $X_t - X_{ab} < \Psi$ or $X_t - X_{rb} < \Psi$

where Ψ is a minimum boundary distance threshold, and we decrease the step size to a small, predetermined value Δt_{min} if either condition is satisfied to ensure the most accurate boundary approximation. When the conditions are not satisfied, we allow the variable step algorithm to adjust the step size according to the traditional SDE adaptive algorithm. This method ensures that the adaptive time stepping methods do not increase the error from the boundary approximations in SHS.

Adaptive Time Stepping Simulation Algorithm

Adaptive time stepping extends the fixed step method by computing error estimates and adjusting the step size before a step is computed. We introduce a new algorithm *ATHMM* that incorporates adaptive time stepping into the *HMM* algorithm. The algorithm tests both error estimates E_b and E_σ, and if either is above an upper threshold, the step size is cut in half. If either error estimate is below a lower threshold, the step size is doubled. If the error is between the two thresholds, then the step size is not adjusted. To avoid large step sizes near boundaries, the algorithm tests the distance to any reflecting or absorbing boundaries and changes the step size to Δt_{min} if it is sufficiently close to a boundary. The following pseudocode describes a single step of *ATHMM*.

Algorithm 4.3: ATHMMSTEP(X_t)

while $\Delta t^2 b'b + J_1^3 \sigma'^2 \sigma > UpperThreshold$

do $\Delta t = \frac{\Delta t}{2}$

while $\Delta t^2 b'b + J_1^3 \sigma'^2 \sigma < LowerThreshold$

do $\Delta t = 2\Delta t$

if $|X_t - X_{ab,\xi}| < 1$ *or* $|X_t - X_{rb}| < 1$

then $\Delta t = \Delta t_{min}$

$k_{t+\Delta t} = X_t - X_{rb} + \frac{1}{2}\left(\sigma W + b(t+\Delta t) + \sqrt{|\sigma|^2 V + (\sigma W + b(t+\Delta t))^2}\right)$

$\Delta k = \max(k_{t+\Delta t}, 0).n - prev\Delta k$

$X_{t+\Delta t} = X_t + b\Delta t + \sigma \Delta W + \sum_{j_1,j_2=1}^{m} L^{j_1}\sigma^{k,j_2} I_{(j_1,j_2)} + n(\Delta k)$

$GuardedProb = max(exp(\frac{-2(n.(X_t - X_{ab,\xi}))(n.(X_{t+\Delta t} - X_{ab,\xi}))}{n.(\sigma\sigma^*(X_t)n)\Delta t}), \xi)$

if $U_1 = rand(0,1) < GuardedProb$

then $FireGuardedTransition$

if $U_2 = rand(0,1) < exp(-\lambda(t - t_{TimeOfLastFire}))$

then $FireProbabilisticTransition$

$t = t + \Delta t$

return (X_t)

Experimental Results

We present the experimental results of our simulation methods to demonstrate the correctness of our models and the performance and accuracy of our methods. We begin by validating the SCD model by comparing the trajectories of the SSA and HEM to demonstrate the correctness of our simulation methods and the modeling methodology. We next validate the biodiesel model using experimental results to compare our simulation results to realistic data. This comparison shows that our modeling and simulation methods virtually match the experimental data.

In this section we also focus on demonstrating the differences between the existing methods (HEM) and our improved simulation methods (HMM). We use the water balance model to demonstrate the improvements of both the absorbing and reflecting boundary

cases. We also use the water balance model to demonstrate the accuracy and performance aspects of using various step sizes. We complete the section by presenting experimental results generated using our ATHMM algorithm for the water balance and Variable Time BioDiesel (VTBD) models that demonstrate the accuracy and efficiency of our methods.

Validation of SCD Using Simulation

To better understand and validate our models, we present SCD simulation results using a variant of the SSA algorithm and the *HEM* algorithm. The SSA simulates chemical reactions consuming reactants and creating products one reaction at a time. Individual reactions in a system are assigned probabilities of occurrence, and probability distributions are used to choose which reaction fires at each iteration. Once a reaction fires, the quantities of reactants and products are updated [51]. The SSA is very accurate because of its level of precision, but it can be inefficient for large systems or fast reactions because many iterations must be completed before results can be observed. To efficiently handle practical systems, computational improvements such as τ-leaping or R-leaping have been devised for the SSA [52, 12]. R-leaping increases the number of reactants consumed and products produced in each step by a factor of R. This increases the efficiency of the approximation, but degrades the accuracy for certain systems.

For simulation of SCD2 and SCD3, we have created a new algorithm, the Hybrid Stochastic Simulation Algorithm *HSSA*, that implements the SSA using R-leaping and discrete transitions between modes. The standard R-leaping SSA is extended to incorporate the discrete dynamics that are found in the SCD2 and SCD3 models. After each iteration of the SSA, the guards for all valid transitions are tested, and a transition that validates its guard conditions is fired if possible. Once the transition resets have been executed, the SSA algorithm resumes in the new state.

One iteration of the SSA describes the evolution of the chemical system over a very small unit of time and is considered to be equivalent to solving the chemical master equation for that state and time step. Interrupting the multiple iterations of the SSA required to define a longer time span does not affect the convergence of the SSA, and is used in all fast/slow chemical simulation techniques for decomposing the problem to make it more efficient.

Therefore, the HSSA can be considered as equivalent to solving the CME with discrete switches.

The SSA algorithm is used to simulate the stochastic time evolution of chemical reactions with high accuracy [12], so we compare the results of our *HSSA* and *HEM* algorithms to demonstrate accuracy of the SHS models and our simulation algorithms. In Figure 21 we compare the sorbitol and fructose concentrations for the SCD1, SCD2, and SCD3, models respectively. We chose sorbitol and fructose because they are the two chemicals that are most directly correlated with the development of cataracts. The initial conditions and parameters for all the experiments are shown in Table 10 from [121]. Figure 21 (a), (b), and (c) display the average concentration at each time step for sorbitol and fructose for 100 runs of the three models. The figures display the comparison between the *HSSA* and *HEM* approximation for sorbitol and fructose to demonstrate the correctness of the SCD models. The 100 *HSSA* simulations completed in 98 hours, and the 100 *HEM* simulations took 8 minutes on a 3GHz desktop computer.

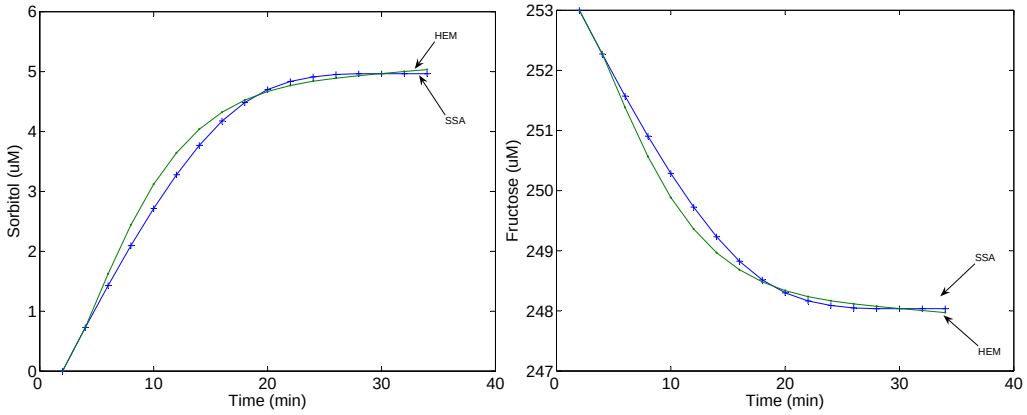
Table 10: Initial conditions and constants for the SCD models

Initial Cond.	Value (μM)
x_1	5.0
x_2	0.0
x_3	5.0
x_4	0.0
x_5	1.0
x_6	253.0
x_7	0.0

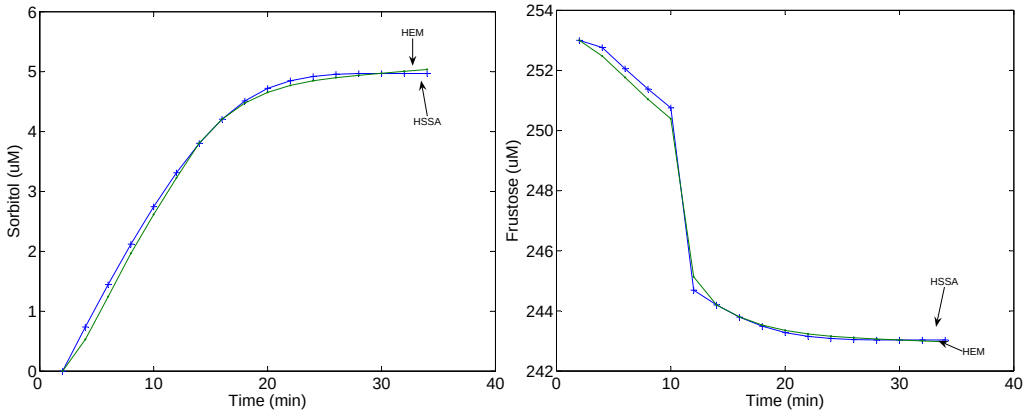
Constant	Value
d_1	10^{-21} μM
d_2	5 μM
d_3	250 μM
d_4	.05
d_5	.05
HEM step	0.0001
R	10

Validation of the Biodiesel Model

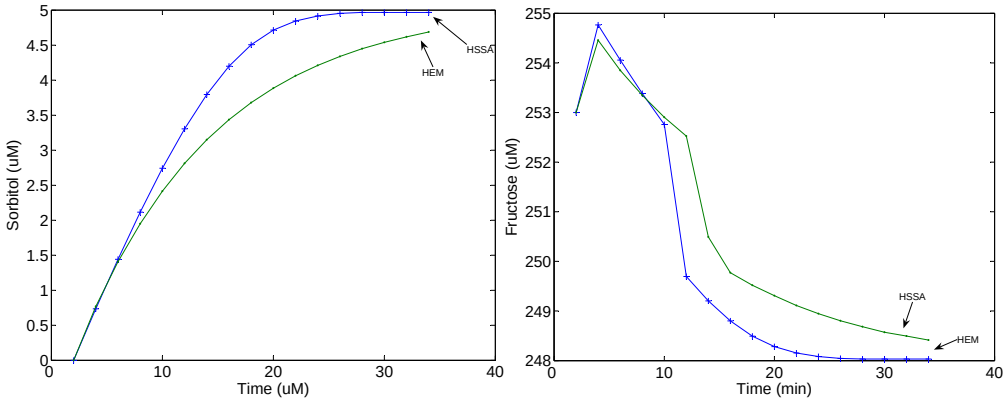
The biodiesel process model is an ideal candidate for validation because experimental data from actual systems is easily gathered and available. However, there are many variations in the types of systems, and experimental data is not available for the exact system we have modeled. Therefore, we have created the Constant Temperature BioDiesel (CTBD) model to more accurately reflect the dynamics of the experimental reactors used to gather the experimental data. Experimental biodiesel reaction systems are designed to isolate as



(a) SCD1



(b) SCD2



(c) SCD3

Figure 21: SCD model validation results

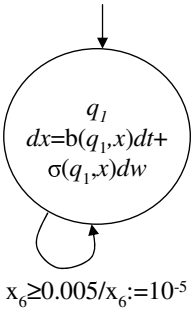


Figure 22: SHS model of the CTBD system

many variables as possible, so the temperature of the system is kept constant. Therefore, we have modified our VTBD model to eliminate temperature fluctuations. The CTBD model has only one discrete state where the temperature is constant. A graphical depiction of the model is shown in Figure 22. The continuous dynamics are kept the same as the VTBD model assuming the temperature is kept constant.

To validate the correctness of our SHS CTBD model, we compare HMM simulation results with the experimental biodiesel system data presented in [108]. Data is available for various mixing methods, but since we only consider a well-mixed reaction, we use their data from the most highly mixed reactions $N_{Re} = 6200$.

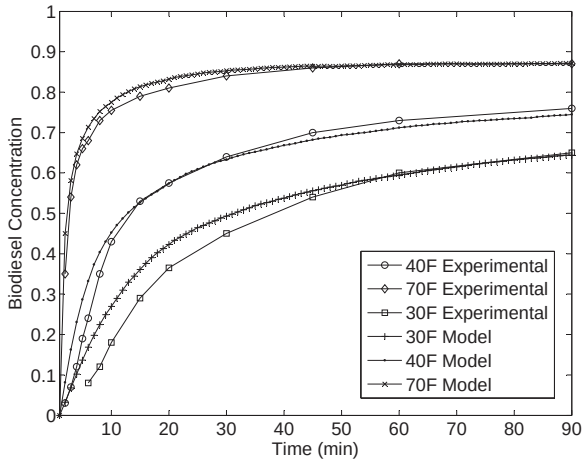


Figure 23: Model and experimental results comparison

A comparison of the experimental results and the results from our model can be seen in Figure 23. We consider three separate temperatures (30 F, 40 F, and 70 F) and we present the experimental results as well as our simulation results for comparison. We present the

difference between the experimental results and the results from our simulation methods in Figure 24. It can be seen that as the simulations progress the accuracy improves implying that the actual mixing in the real reactor is not ideal as we have assumed in our model.

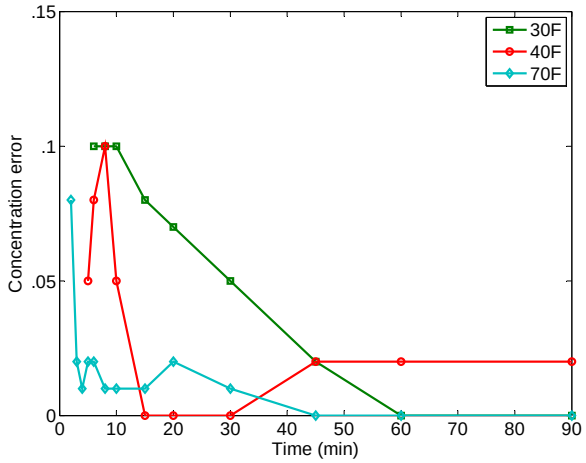


Figure 24: Error of simulated CTBD model

Absorbing Boundary Crossing Detection

To compare differences between boundary crossing detection methods, we compared simulations of *HEM* and *HMM* for the water balance model. We used the same Brownian motion for each set of simulation comparisons to highlight the algorithmic differences. In Figure 25 the switching electrolyte boundary is presented for the *HEM* algorithm and the *HMM* algorithm, and the difference between the detection times is shown by the gap between the indicated detection points. The advanced method anticipates the boundary crossing using stochastic methods to avoid error incurred by over-shooting the crossing, while the *HEM* method restarts the process only after the crossing is detected. It is evident that the anticipatory methods of the improved technique significantly alter the resulting trajectory, thereby reducing the error incurred. In the water/electrolyte system this may mean that the actual system may react sooner than the traditional simulation of the model predicts. For these simulations we used a step size of $\Delta t = .05$ and initial conditions shown in Table 11.

We consider the ADH concentration to demonstrate the reflecting boundary algorithm differences between the *HEM* algorithm and the *HMM* algorithm. In Figure 26 the reflecting

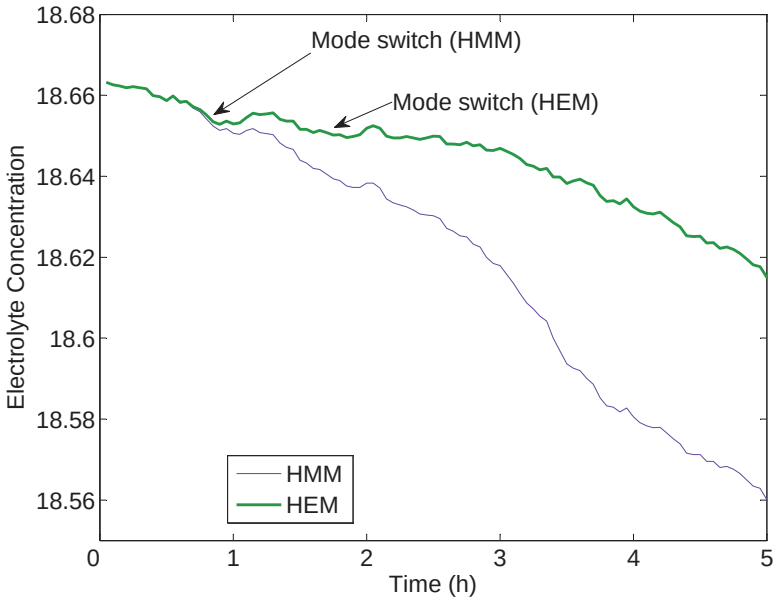


Figure 25: Absorbing boundary in the water balance model

Table 11: Initial conditions for water balance model

Variable	x_1	x_2	x_3
Absorbing	39790	2132	1
Reflecting	39700	2132	11

boundary is represented by the dark line at $ADH = 12$. Using the *HEM* algorithm, the trajectory reaches the boundary and is kept within the valid state. However, the dynamics of the actual system are not accurately represented because the real system reaches a reflecting saturation level at the boundary. Using the *HMM* method, the trajectory highlights the stochastic effect of the reflected saturation boundary. The receptors in the real system cannot maintain the full concentration at 12 because the molecules of ADH have to be released to permit new molecules to bind. The receptors cannot fire in an unbound state, so the influence of the ADH concentration must be reduced, as is evidenced by the small drops in concentration near the boundary. These drops eventually lead to a distinct difference between the outcomes of the two trajectories. While both trajectories eventually reach an equilibrium (not included in the figure), the difference in the dynamics leading to equilibrium may reveal new insights into the system. For these simulations, we used a step size of $\Delta t = .05$ and initial conditions shown in Table 11.

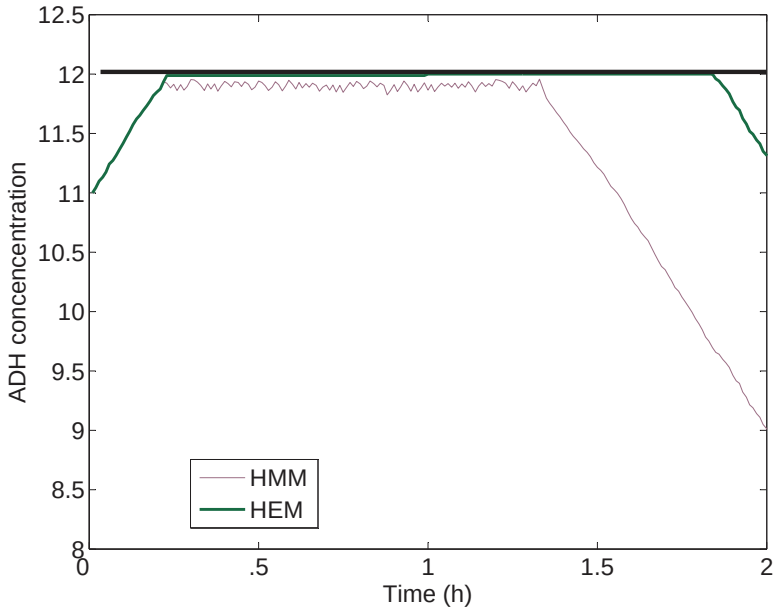


Figure 26: Reflecting boundary in the water balance model

Performance results for the fixed step implementations are presented in Table 12. We ran 1000 sequential simulations of each algorithm at the given resolution. The *HMM* algorithm increases the running time relative to the *HEM* method; however, the increase is small, the method scales well, and the accuracy improvement is significant. The simulations were performed on a 3GHz desktop computer with 1GB of RAM.

Table 12: Execution times at various resolutions for the water balance model

Resolution	*HEM* (sec)	*HMM* (sec)
.0001	352	374
.0002	176	189
.0005	70	75
.001	37	38

Adaptive Time Stepping

We present experimental results using our ATHMM algorithm for the water balance and VTBD models. We demonstrate the accuracy of the ATHMM method near boundaries by comparing it to the HMM method using the water balance model. We also demonstrate the accuracy and efficiency of the ATHMM algorithm using the VTBD model.

Water Balance Model The step size of the approximation directly influences the accuracy of the approximation, so we use the water balance model to demonstrate the performance of the adaptive time stepping algorithm. In Figure 27, we compare four different step sizes of the *HMM* algorithm and resulting trajectories. We used the Wiener process from the highest-resolution trajectory with each lower-resolution simulation to ensure the comparison is accurate. Using more accurate approximation techniques with higher orders of convergence ensures that larger time steps can be used to maintain acceptable accuracy.

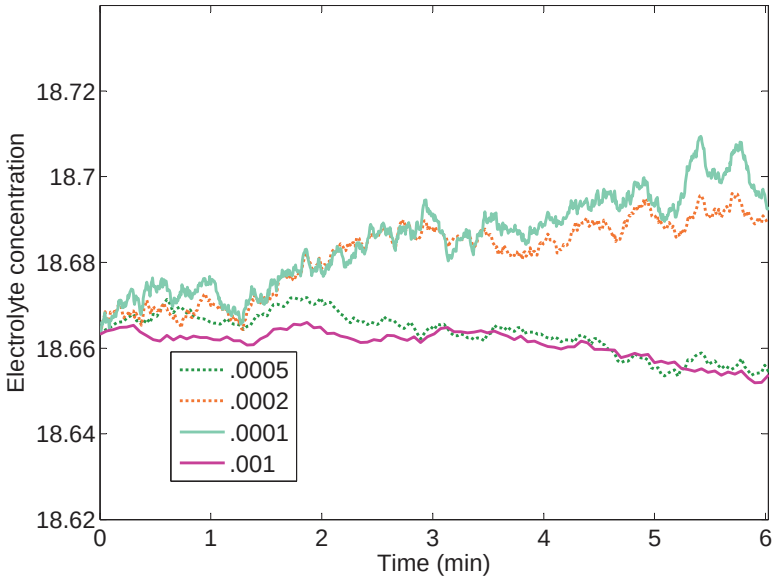


Figure 27: Step size comparison for the water balance model

We compare simulations of the *HMM* algorithm (with time step $\Delta t = .01$) with the adaptive time stepping *ATHMM* algorithm in Figure 28. It can been seen in the figure that as the trajectory becomes less steep, the step size increases to increase efficiency, and when the trajectory crosses the threshold near the boundary, the variable time stepping algorithm uses the highest resolution step to accurately estimate the boundary crossings. The fixed step implementation required 1,000 steps for the approximation while the variable step method with step error bounds of $|E_b|+|E_\sigma| < .01$ required only 295 steps, over tripling the efficiency. The figure shows that the accuracy is not significantly hindered by using the variable step size algorithm. Tighter error bounds can be used to create more accurate approximations at the cost of efficiency.

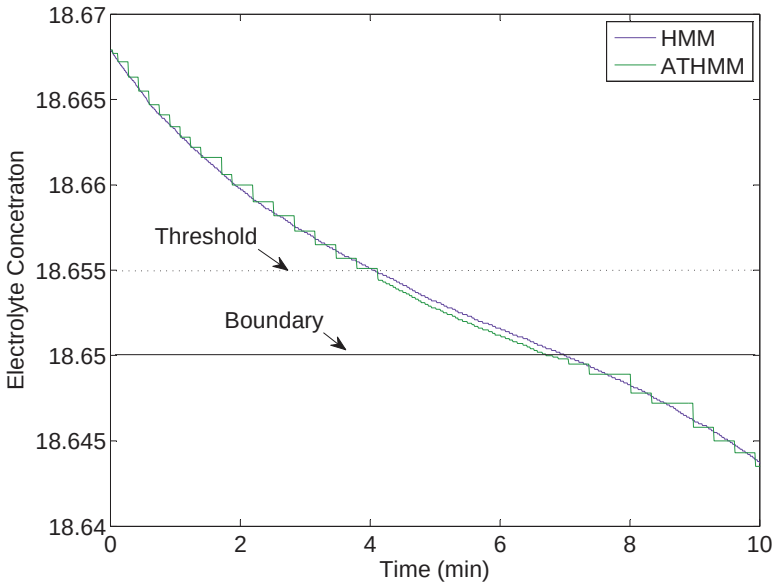


Figure 28: Variable step example of the water balance model

VTBD Model We use the VTBD model to demonstrate the performance and accuracy advantage of using adaptive time stepping methods. We consider the VTBD model with initial conditions: $x_1 = 0$, $x_2 = .8$, $x_3 = 3$, $x_4 = 0$, $x_5 = 10$, $x_6 = 0$, and $x_7 = 70$. We used the HMM algorithm with time steps: $\Delta t = .001, .0001, .00001, .000001$. The ATHMM algorithm requires upper and lower error bounds, so we used the following bounds: $UpperThreshold = .01, .001, .0001, .00001$ and $LowerThreshold = .0001, .00001, .000001, .0000001$. We present the execution times and resulting overall error estimates for the fixed and adaptive methods in Figure 29. It can be seen in the figure that the adaptive time step methods provide significant accuracy and efficiency improvements over fixed time step methods.

Summary

In this work we present the individual components of SHS simulation methods along with several complete SHS simulation algorithms. We describe two methods for simulating SDEs as well as traditional and advanced methods for simulating switching and reflecting boundaries. We also present a method for simulating probabilistic transitions and an adaptive time stepping algorithm for SHS. Our advanced SHS simulation methods presented in this chapter demonstrate an improvement over previous methods in both accuracy and efficiency.

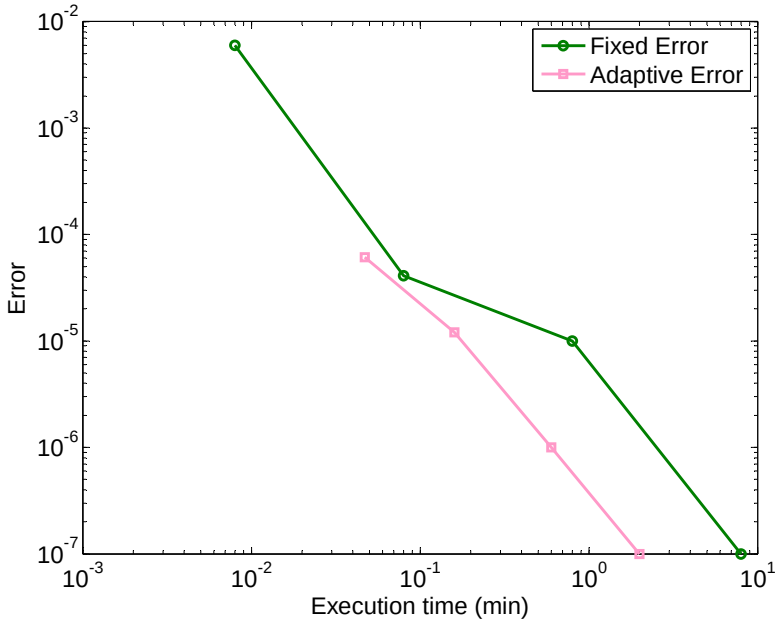


Figure 29: Error comparison of time stepping methods for the VTBD model

SHS simulation methods that improve efficiency and accuracy are important to develop to improve the utility of SHS models. In this chapter we compare the results of our SHS simulation method with simulations of the Stochastic Simulation Algorithm (SSA) for the SCD models to demonstrate the correctness of our methods. We also compare our SHS simulation method with experimental results for the biodiesel model to show the accuracy of both the simulation method and the model. We also demonstrate the improvements of our advanced SHS simulation algorithm (HMM) by comparing various boundary scenarios for the water balance model. We demonstrate improvements in both the switching and reflective boundaries by comparing the HEM and HMM algorithms. We also present results of step size comparisons and a demonstration of the ATHMM algorithm and the related efficiency improvements.

CHAPTER V

VERIFICATION OF SHS

Simulation methods can determine the outcome of individual trajectories of a model, but they are not sufficient to verify the behavior of a system. Therefore, advanced analysis methods such as verification are useful for determining properties such as reachability or safety for a system. Exhaustive verification methods can be used to determine these properties for any initial state of the system [81]. Verification of reachability properties for Stochastic Hybrid Systems (SHS) aims at determining the probability that the system will reach a set of desirable or unsafe states. Reachability analysis is an important problem because it provides a formal framework to analyze complex, realistic systems that operate in the presence of uncertainty and variability [83]. Verification of reachability properties for such systems is a critical problem because the results of the verification can help scientists gain insights into the modeled systems and expose flaws or validate the intended function.

Developing sound computational methods for verification is challenging because of the interaction between the discrete and the continuous stochastic dynamics. A formal analysis framework is necessary to ensure these dynamics are handled properly. Further, computational restrictions of exhaustive verification can limit the size of the system that can be handled, so efficient, scalable methods are required [25].

In this chapter we present a dynamic programming verification method for SHS that utilizes Markov decision processes and a value iteration technique for analysis [83]. We also present a parallel decomposition of the algorithm that can be used to scale the analysis method for significantly large and realistic systems. While the method is constrained by the curse of dimensionality, we show that it is useful for realistic systems [120].

We also present experimental results for four different systems to demonstrate the flexibility and applicability of the analysis method. We present a navigation benchmark [40], a room heating benchmark [40], the SCD models [120], and the Variable Temperature BioDiesel (VTBD) model [119]. We present reachability or safety analysis results for each system as well as performance results to show the methods applicability to realistic systems.

Stochastic Verification

Safety is a special case of the reachability problem, so we formulate the reachability problem first and then the safety problem. The target and unsafe sets for SHS are unions of target and unsafe sets respectively for multiple modes. Let $T = \cup_{q \in Q_T} \{q\} \times T^q$ and $U = \cup_{q \in Q_U} \{q\} \times U^q$ be subsets of S representing the set of target and unsafe states respectively. We assume that T^q and U^q are proper open subsets of X^q for each q, i.e. $\partial T^q \cap \partial X^q = \partial U^q \cap \partial X^q = \emptyset$ and the boundaries ∂T^q and ∂U^q are sufficiently smooth. We define $\Gamma^q = X^q \setminus (\bar{T}^q \cup \bar{U}^q)$ and $\Gamma = \cup_{q \in Q} \{q\} \times \Gamma^q$. The initial state (that, in general, can be a probability distribution) must lie outside the sets T and U. The transition measure $R(s, A)$ is assumed to be defined so that the system cannot jump directly to U or T.

Consider the stopping time $\tau = \inf\{t \geq 0 : s(t) \in \partial T \cup \partial U\}$ corresponding to the first hitting time of the boundary of the target or unsafe set. Let s be an initial state in Γ, then we define the function $V : \bar{\Gamma} \to \mathbb{R}_+$ by

$$V(s) = \begin{cases} E_s[I_{(s(\tau^-) \in \partial T)}], & s \in \Gamma \\ 1, & s \in \partial T \\ 0, & s \in \partial U \end{cases}$$

where E_s denotes the expectation of functionals given the initial condition s and I denotes the indicator function. The function $V(s)$ can be interpreted as the probability that a trajectory starting at s will reach the set T while avoiding the set U. If the state hits the boundary of the unsafe or target set, then the value function takes the value 0 and 1 respectively and it is assumed that the execution of the SHS terminates.

Given the assumptions on the sets T and U and their boundaries, we can construct a bounded function $c : \bar{S} \to \mathbb{R}_+$ continuous in x such that

$$c(q, x) = \begin{cases} 1, & \text{if } x \in \partial T^q \\ 0, & \text{if } x \in \partial U^q \cup \partial X^q \end{cases} .$$

We define a counting process p^* by

$$p^*(t) = \sum_{i=1}^{\infty} I_{(t \geq t_i)} I_{(s(t_{i^-}) \in \partial S)}.$$

The process $p^*(t)$ counts the number of times the trajectory hits the boundary ∂S and jumps up to time t [33]. Then, the value function V can be written as

$$V(s) = E_s \left[\int_0^{\infty} c(q_{t^-}, x_{t^-}) dp^*(t) \right]. \tag{6}$$

The formulation of the reachability problem described above can be modified to describe safety. In a safety problem, we are given a set of safe states and we want to compute the probability that the system execution from an arbitrary (safe) initial state will go outside the safe set. Let $B = \cup_{q \in Q_B} \{q\} \times B^q$ be a subset of S representing the set of safe states. We assume that the set of unsafe states $X^q \setminus B^q$ for each q is a proper subset of X^q, i.e. $\partial X^q \cap \partial B^q = \emptyset$. The initial state must lie inside the safe set B and the transition measure $R(s, A)$ is defined so that the system cannot jump out of the safe set directly to the unsafe set. We can transform the safety problem to a reachability problem by defining the target set as $T^q = X^q \setminus B^q$ and the unsafe set as $U^q = \emptyset$. Note that in this case, the definition of Γ^q becomes $\Gamma^q = X^q \setminus (T^q \cap U^q) = B^q$. Clearly with this transformation, the probability that the system is unsafe can be computed as the value function described by (6) similarly to the reachability problem [84].

Using a dynamic programming argument, it can be shown that the value function V for the reachability problem of stochastic hybrid systems is similar to the value function for the exit problem of a standard stochastic diffusion, but the running and terminal costs depend on the value function V itself. A detailed proof of the derivation can be found in [85]. We define $L^V(q, x) = \lambda(q, x) \int_\Gamma V(y) R((q, x), dy)$, $\psi^V(q, x) = c(q, x) + \int_\Gamma V(y) R((q, x), dy)$, and $\Lambda(t) = \exp\left\{ - \int_0^t \lambda(q_0, x_z) dz \right\}$. Then for $s \in \Gamma$

$$V(s) = E_s \left[\int_0^{t_1^*} \Lambda(t) L^V(q_{t^-}, x_{t^-}) dt + \Lambda(t_1^*) \psi^V(q_{t_1^*}, x_{t_1^*}) \right]. \tag{7}$$

Equation (7) is similar to the discounted cost criterion with a target set of a standard stochastic diffusion [87]. The main difference is that the running cost $L^V(q,x)$ and the terminal cost $\psi^V(q,x)$ depend on the value function. It should be noted that the SHS satisfies the strong Markov property, and the same procedure can be repeated every time a jump occurs. Further, it can be shown under the non-degeneracy assumption that V is bounded and continuous [85]. Then, based on the results of [87] V can be characterized as the viscosity solution of a system of Hamilton-Jacobi-Bellman equations. In particular, V is the unique viscosity solution of the system of equations

$$\mathcal{H}_V\left((q,x),V,D_xV,D_x^2V\right)=0 \text{ in } \Gamma^q,\ q\in Q \tag{8}$$

with boundary conditions

$$V(q,x)=\psi^V(q,x) \text{ on } \partial\Gamma^q,\ q\in Q \tag{9}$$

where

$$\mathcal{H}_V\left((q,x),V,D_xV,D_x^2V\right)=b(q,x)D_xV+\frac{1}{2}\text{tr}(a(q,x)D_x^2V)+\lambda(q,x)V+L^V(q,x).$$

Equation (8) describes a set of coupled second-order partial differential equations (one for each discrete state), with boundary conditions given by (9), that can be viewed as a set of HJB equations associated with the reachability problem for the SHS. The coupling between the equations arises because the value function in a particular mode depends on the value function in the adjacent modes and is formally captured by the dependency of the running and terminal costs $L^V(q,x)$ and $\psi^V(q,x)$ on the value function V.

Numerical Methods Based on Dynamic Programming

One of the advantages of characterizing reachability as a viscosity solution is that for computational purposes we can use well known numerical algorithms. In this paper we employ the finite difference method presented in [87] to compute locally consistent Markov Chains (MC) that approximate the original stochastic process while preserving local mean

and variance. We consider a discretization of the state space denoted by $\bar{S}^h = \cup_{q\in Q}\{q\}\times\bar{S}_q^h$ where $\bar{S}_q^h$ is a set of discrete points approximating X^q and $h > 0$ is an approximation parameter characterizing the distance between neighboring points. By abuse of notation, we denote the sets of boundary and interior points of $\bar{S}_q^h$ by ∂S_q^h and S_q^h respectively. By the boundness assumption, the approximating Markov chain has finitely many states that are denoted by $s_n^h = (q_n^h, \xi_n^h),\ n = 1, 2, \ldots, N$.

First, we consider the continuous evolution of the SHS between jumps and assume that the state is (q, x). The local mean and variance given by the Stochastic Differential Equation (SDE) (3) on the interval $[0, \delta]$ are

$$E[x(\delta) - x] = b(q, x)\delta + o(\delta)$$
$$E[(x(\delta) - x)(x(\delta) - x)^T] = a(q, x)\delta + o(\delta).$$

Let $\{q_n^h = q, \xi_n^h\}$ describe the MC on $S_q^h \subset X^q$ with transition probabilities denoted by $p_D^h((q, x), (q', x'))$. A locally consistent MC must satisfy

$$E[\Delta\xi_n^h] = b(q, x)\Delta t^h(q, x) + o(\Delta t^h(q, x))$$

and

$$E[(\Delta\xi_n^h - E[\Delta\xi_n^h])(\Delta\xi_n^h - E[\Delta\xi_n^h])^T] = a(q, x)\Delta t^h(q, x) + o(\Delta t^h(q, x))$$

where $\Delta\xi_n^h = \xi_{n+1}^h - \xi_n^h, \xi_n^h = x$ and $\Delta t^h(q, x)$ are appropriate interpolation intervals (or the "holding times") for the MC.

The diffusion transition probabilities $p_D^h((q, x), (q', x'))$ and the interpolation intervals can be computed systematically from the parameters of the SDE (details can be found in [87]). For a uniform grid with e_i denoting the unit vector in the i^{th} direction, the transition probabilities are

$$p_D^h((q, x), (q, x \pm he_i)) = \frac{a_{ii}(q, x)/2 + hb_i^{\pm}(q, x)}{Q(q, x)}$$

$$p_D^h((q,x),(q,x+he_i+he_j)) = p_D^h((q,x),(q,x-he_i-he_j)) = \frac{a_{ij}^+(q,x)}{2Q(q,x)}$$

$$p_D^h((q,x),(q,x-he_i+he_j)) = p_D^h((q,x),(q,x+he_i-he_j)) = \frac{a_{ij}^-(q,x)}{2Q(q,x)}$$

and the interpolation intervals are $\Delta t(q,x) = h^2/Q(q,x)$ where

$$Q(q,x) = \sum_i a_{ii}(q,x) - \sum_{i,j:i\neq j} \frac{|a_{ij}(x)|}{2} + \sum_i h|b_i(q,x)|$$

, and $a^+ = \max\{a,0\}$ and $a^- = \max\{-a,0\}$ denote the positive and negative parts of a real number.

Next, we consider the jumps with transition rate $\lambda(q,x)$ and transition measure $R((q,x),A)$. Suppose that at time t the state is $\{q_n^h = q, \xi_n^h = x\}$. The probability that a jump will occur on $[t,t+\delta)$ conditioned on the past data can be approximated by

$$P[(q,x) \text{ jumps on } [t,t+\delta) | q(s), x(s), w(s), s \leq t] = \lambda(q,x)\delta + o(\delta).$$

The i^{th} jump of the approximating process is denoted by $\zeta((q,x),\rho_i)$ where ρ_i are independent random variables with distribution $\bar{R} = \{\rho : \zeta((q,x),\rho_i) \in A\} = R((q,x),A)$ with compact support Π. Let ζ^h be a bounded measurable function such that $|\zeta^h((q,x),\rho) - \zeta((q,x),\rho)| \to 0$ as $h \to 0$ uniformly in x for each ρ and that satisfies $\zeta^h((q,x),\rho) \in \bar{S}^h$.

If $x \in S_q^h$, then with probability $p_{jump}^h(q,x) = \lambda(q,x)\Delta t^h(q,x) + o(\Delta t^h(q,x))$ there is a jump and the next state is $(q_{n+1}^h, \xi_{n+1}^h) = \zeta^h((q,x),\rho_i)$ and with probability $1 - p_{jump}^h(q,x)$ the next state is determined by the diffusion probabilities p_D^h, thus the transition probabilities are given by

$$p^h((q,x),(q',x')) =$$
$$(1 - p_{jump}^h(q,x))p_D^h((q,x),(q',x')) + p_{jump}^h(q,x)\bar{R}\{\rho : \zeta^h((q,x),\rho) = (q',x'-x)\}. \quad (10)$$

For the points $x \in \partial S_q^h$ in the boundary, the next state is determined by $\zeta^h((q,x),\rho_i)$ with

probability 1 and the transition probabilities are given by

$$p^h((q,x),(q',x')) = \bar{R}\{\rho : \zeta^h((q,x),\rho) = (q', x' - x)\} \tag{11}$$

Let $\bar{T}^h = \bar{S}^h \cap \bar{T}$ and $\bar{U}^h = \bar{S}^h \cap \bar{U}$ denote the discretized target and unsafe sets respectively. We denote by n_i the times of the jumps between modes and ν_h the stopping time representing that $(q_n^h, \xi_n^h) \in \bar{T}^h \cup \bar{U}^h$, then the value function V can be approximated by

$$V^h(s) = E_s \left[\sum_{n=0}^{\nu_h} c(q_n^h, \xi_n^h) I_{(n=n_i)} \right].$$

The function V^h can be computed using a value iteration algorithm. We initialize the value function $\tilde{V}_0^h(q,x) = 0$ for every (q,x), then the dynamic programming iteration is given by

$$\tilde{V}_{n+1}^h(q,x) = \left[\sum_{q',x'} \tilde{p}^h((q,x),(q',x')) \tilde{V}_n^h(q',x') \right]$$

This converges to the value function V of the SHS as $h \to 0$. The proof of the convergence can be found in [85].

Analysis of the computational complexity of value iteration algorithms is usually based on the contraction property of the iteration operator. The iteration operator used for verification of SHS corresponds to an undiscounted criterion and showing that it is a contraction mapping is more involved. We have proved that the iteration operator restricted to an appropriate set is a contraction mapping with respect to some weighted infinity norm and the polynomial-time complexity of the algorithm [85]. Reachability analysis of stochastic hybrid systems is polynomial on the number of states of the approximating Markov process; however, this number grows exponentially with the dimension of the continuous state space. Therefore, application of the approach is limited to low dimensional systems. Although scalability is a limiting factor, using parallel methods make the approach feasible for realistic systems.

Parallel Decomposition of the Verification Algorithm

Storing the values for the value iteration algorithm for even reasonably-sized models

requires several gigabytes of memory, so we have developed a parallel value iteration implementation to improve the scalability of the algorithm. Parallel algorithms traditionally cannot take full advantage of the increased computing capabilities because slow intra-computer communication delays computation. Some algorithms require more communication than others, and increased communication further decreases efficiency of the algorithm. Therefore, algorithms that minimize communication maximize parallel algorithm performance.

Dynamic programming algorithms are very natural to parallelize because of the repetitive nature of the algorithms and the minimal communication required; however, care must be taken to ensure that the algorithm converges to the correct solution in a parallel implementation. The value iteration algorithm is guaranteed to converge in a parallel implementation as long as communication between the partitions happens periodically [19]. The discrete switches in our system affect the structure of the state space, but do not change the overall convergence results. The way the state space is partitioned directly affects the efficiency of the parallel method, so the partitioning must be chosen carefully to minimize communication required.

To partition the problem for multiple processors we divide the state space in half p times into 2^p equally-sized partitions and assign each partition to a processor [4]. A graphical depiction of the decomposition can be seen in Figure 30. Each partition is then executed independently and the values at the boundaries that are shared with other partitions are periodically updated to guarantee convergence to the solution. The processors routinely calculate the collective amount of change to determine when value iteration can complete. We use a parallel communication formalism Message Passing Interface (MPI) to execute the communication between processors to ensure efficient communication.

For all of our parallel experiments we used $p = 5$ for simplicity. To divide the state space in an effort to minimize communication required, we choose the five largest dimensions and split each into two parts. The first split defines 2 partitions and 1 communication boundary, the second defines a total of 4 partitions and 4 communication boundaries, the third defines a total of 8 partitions and 12 communication planes, the fourth defines a total of 16 partitions and 32 communication hyperplanes, and the fifth defines all 32 partitions

[4] p must be less than or equal to the number of continuous dimensions of the model

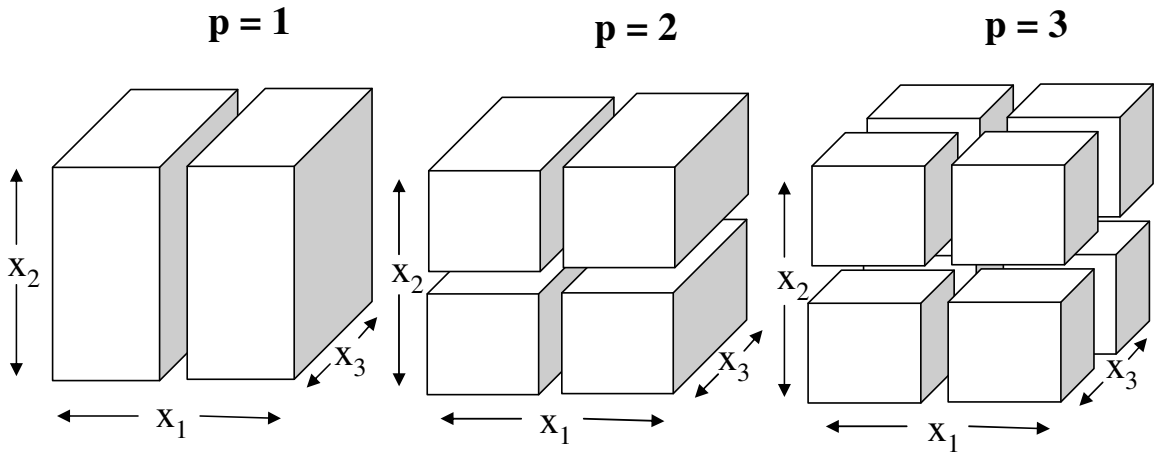


Figure 30: Parallel decomposition of the state space

and 80 communication hyperplanes. The 32 processors are assigned to the 32 partitions to minimize the required communication to minimize the overhead required by the technique. More processors could be used to further enhance the scalability of this technique, but we found that 32 is an adequate number for this size of a problem.

Experimental Results

In this section we present experimental results for several SHS models to demonstrate the usability and performance of the exhaustive verification algorithm. We begin by describing the navigation benchmark which is a four dimensional system eight discrete modes [40]. It is a reasonably-sized system that can be verified without parallel methods.

The next system we consider is the room heater benchmark from [40]. The model is a three dimensional model with twelve discrete modes; however, it can easily be expanded to incorporate more continuous dynamics and discrete modes. This model allows us to demonstrate both the simple and parallel versions of our algorithm.

We also present experimental results for the SCD models. We compare the differences between the value functions of these models to examine both the safety and reachability problems. Because of the size and complexity of the systems, we use parallel methods to perform the analysis.

Our last model that we verify is the biodiesel model. This model is also large and complex, so we use parallel methods to analyze the system. We present the results of this

analysis because they are used in the next chapter to compare to the results of our Monte Carlo-based verification methods.

Navigation Benchmark

We first illustrate our approach using a stochastic version of the navigation benchmark presented in [40] and used in [83]. The benchmark describes an object moving within a bounded 2-dimensional region partitioned into cells X^q, $q \in \{0, 1, \ldots, N_c\}$ as shown in Figure 31. Let $x = [x_1, x_2]^T$ and $v = [v_1, v_2]^T$ denote the position and the velocity of the object respectively. The behavior is defined by the ODE $\dot{v} = A(v - v_d^q)$ where $A \in \mathbb{R}^{2\times 2}$ and $v_d^q = [sin(q\pi/4), cos(q\pi/4)]^T$. Selecting the matrix A and adding a diffusion term, the dynamics of the object are described by the SDE

$$dx = (\tilde{A}x + \tilde{B}u_d^q)dt + \Sigma dw$$

where $x = [x_1, x_2, v_1, v_2]^T$, $u_d^q = [0, 0, v_d^q]^T$, $w(t)$ is an $\mathbb{R}^4$-valued Wiener process,

$$\tilde{A} = \begin{bmatrix} 0 & I_2 \\ 0 & A \end{bmatrix}, \; A = \begin{bmatrix} -1.2 & 0.1 \\ 0.1 & -1.2 \end{bmatrix}, \text{ and } \Sigma = 0.1 I_4.$$

Consider the target set T and the unsafe set U shown in Figure 31. Given an initial state $s_0 = (q_0, x_0)$, we want to compute the probability that the state will reach T while avoiding U. Figure 31 also shows sample trajectories. In order to apply the approach described in this paper, we under-approximate each cell X^q by $\tilde{X}^q$ by considering a smooth boundary $\partial\tilde{X}^q$. We also define a transition measure $R((q, x), A)$ so that the state jumps into an adjacent cell if it hits an "inner" boundary and jumps into the same cell if it hits an "outer" boundary. The transition rate is assumed to be zero. We discretize the state space using a uniform grid with approximation parameter $h > 0$ and apply the verification method to compute $V^h(q, x)$. As $h \to 0$, $V^h(q, x)$ converges to the solution $V(q, x)$ of the stochastic approximation of the benchmark problem.

Since the continuous state space of the example is 4-dimensional, we select to plot a projection of V^h for initial velocity $v_0 = [0, 0]^T$. Figure 32 shows this projection for $h = 0.1$

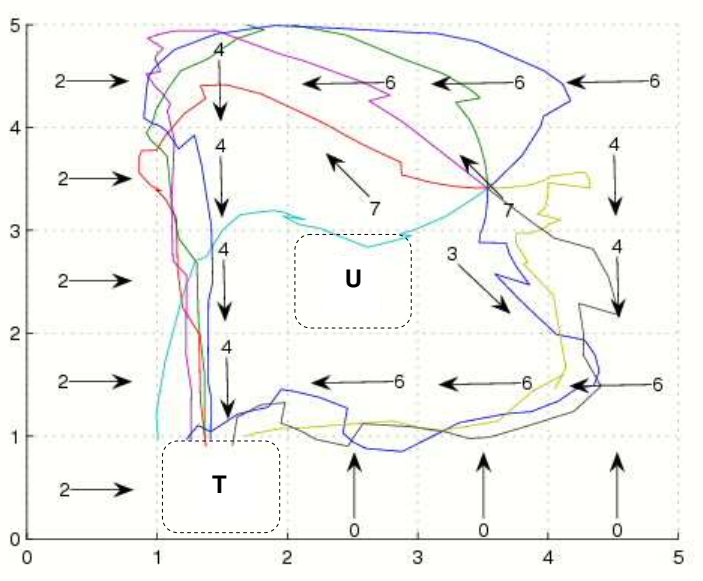


Figure 31: The navigation benchmark state space

Table 13: Performance data for the navigation benchmark

h	Time (minutes)	Number of States
.5	.5	2500
.25	7	32400
.1	200	1147041
.05	5110	17147881

that describes the probability that a trajectory starting from $(q, [x_1, x_2, 0, 0]^T)$ will reach T while avoiding U. The computational performance of the algorithm is illustrated in Table 13. All data was collected using a 3.0 GHz desktop computer with 1 GB RAM and they are consistent with the polynomial-time complexity of the algorithm.

Room Heater Benchmark

A modeling benchmark of a room heating problem has been presented for a simple three room system in [40]. The benchmark models the temperature dynamics of a building with three rooms and two mobile heaters. The temperature in each room x_i depends on the temperature of the adjacent rooms, the outside temperature u, and whether a heater is in the room and turned on.

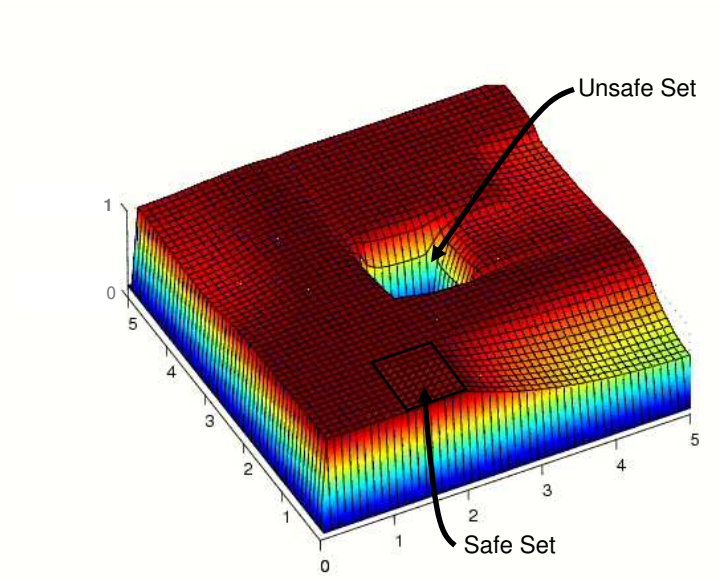


Figure 32: Value function for the navigation benchmark

We have generated a stochastic version of the benchmark. The SDE describing the continuous dynamics of the system is

$$dx = (Ax + Bu + Cq)\,dt + \Sigma dw$$

where

$$A = \begin{bmatrix} -.9 & .5 & 0 \\ .5 & -1.3 & .5 \\ 0 & .5 & -.9 \end{bmatrix},\ B = \begin{bmatrix} .4 \\ .3 \\ .4 \end{bmatrix},$$

$C = diag(6, 7, 8)$, $u = 4$, $\Sigma = diag(0.1)$, q is a vector consisting of 0's and 1's representing the position and state of the heaters, and $w(t)$ is an $\mathbb{R}^3$-valued Wiener process.

The discrete states of the system describe the position and condition of the heaters in the rooms. If a heater is in a room and on, then a one is placed in the corresponding position of that room. If the heater is not in the room or is in the room but off, then a zero is placed in the corresponding position. The heating benchmark has twelve heater modes as shown in Figure 33. Mode transitions are denoted by the arcs between nodes and are defined using a control policy for moving the heater. The control policy is captured by the invariants of the discrete states. We consider the following control policy for rooms i and

j. If a heater is present in room i, but off, it is switched on if $x_i \leq 19$ and a heater that is on is switched off if $x_i \geq 20$. A heater is moved from room j to an adjacent room i if the following conditions are true: (i) room i is without a heater, (ii) room j currently has a heater, (iii) $x_i \leq 17$, and (iv) $x_j - x_i \geq 1$.

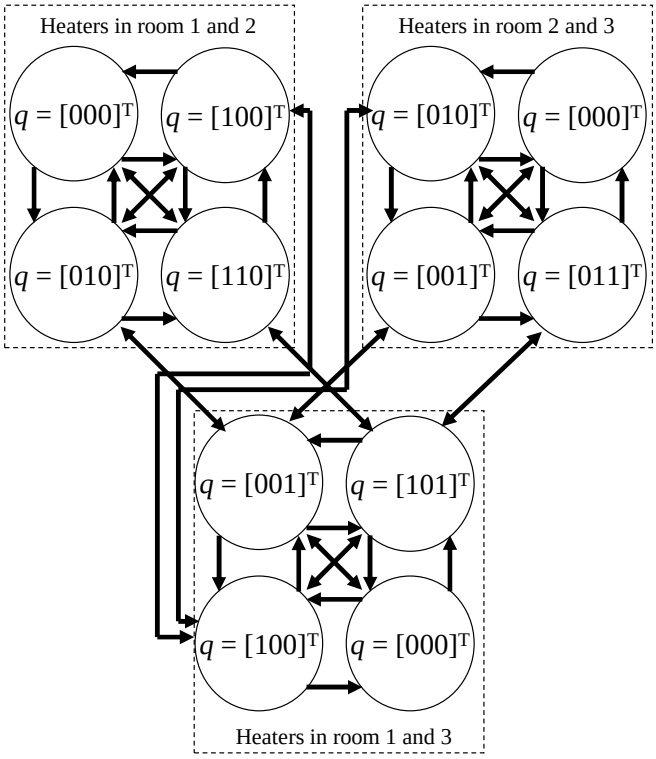


Figure 33: Automaton for the room heater benchmark

We discretized the continuous state space by assuming that the safe set is described by $x_i = (10, 20), i = 1, 2, 3$ and the approximation parameter is set to $h = 0.25$. Since there are 12 discrete modes, the number of states of the approximating process is $12 \times 42^3 = 889056$ (including the boundary of the safe set). The room heater benchmark evolves in a three-dimensional continuous state space, hence it is difficult to visualize the value function. To illustrate our results, we have set a pre-defined threshold (0.1) that describes the acceptable probability for reaching the unsafe set. Then, for each initial mode we plot the "safe" set as the set of states that have a probability below the threshold to reach the unsafe set. Figure 34 shows the safe set. The iterative algorithm executed in approximately 49 minutes on a 3.0 GHz desktop computer.

An important characteristic of the room heater benchmark is that it can be easily

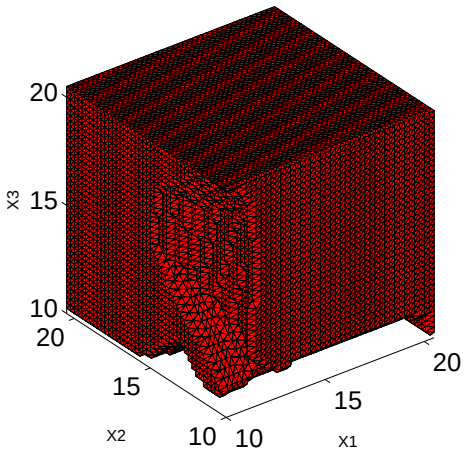


Figure 34: Room heater benchmark safe states for $q = [110]^T$

scaled up to an arbitrary number of rooms that determine the dimension of the continuous state space. Using parallel methods for dynamic programming [19], we have verified a 6-dimensional version of the room heater benchmark. For approximation parameter $h = 0.25$, the discrete approximation is verified in approximately 10^3 min in a high-performance computer cluster with 4 processors.

Sugar Cataract Development

In this section we present the implementation details and the results of the verification of the SCD models from Chapter *III*. We also describe performance characteristics of the system at various resolutions of the state space.

Biologists have determined that a ratio of sorbitol to fructose that is greater than one is correlated to the beginning stages of sugar cataract formation [13]. It has been shown that fructose and SDH play a significant role in the accumulation of sorbitol in the eye, which in turn begins the formation of sugar cataracts.

Simulations can determine whether or not a certain starting state will eventually lead to sugar cataract formation; however, it is much more useful to examine all possible starting states, which can be accomplished through verification. Since a ratio of sorbitol to fructose that is greater than one is correlated to the beginning stages of sugar cataract formation, we have identified those states that meet that criteria as the set of unsafe states. States

with a sorbitol to fructose ratio greater than 0.5 but less than 1 correspond to eyes that are possibly at risk, but not at high risk of cataract formation. States with a ratio of less than 0.5 are at low risk, and therefore, are desirable, or target states. The unsafe and target sets are depicted in Figure 35.

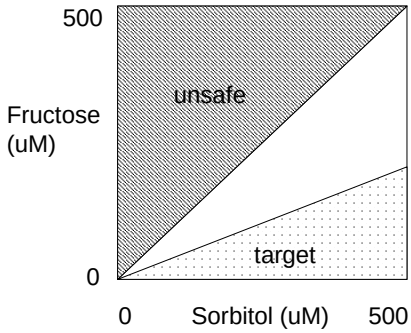


Figure 35: Graphical depiction of the unsafe and target sets

Medication can be administered to inhibit the enzyme SDH, which is intended to help keep sugar cataracts from forming in high risk patients. The safety probability for the medicated model describes the probability that an initial state will transition to an unsafe state given the administration policy for the drug. The reachability probability for a state describes the probability that the patient will transition from the current state to a state without first reaching the unsafe states under the given drug administration policy.

The SCD1, SCD2, and SCD3 models are implemented using the constants presented in Table 10. In order to apply the approach described in this paper we under-approximate each discrete region X^q by $\tilde{X}^q$ by considering a smooth boundary $\partial\tilde{X}^q$. The discrete approximations must be created using the finest resolution possible to ensure an accurate result, but increasing the fineness of the resolution causes a significant increase in the number of states, so a balance of accuracy and efficiency must be found. Using scaling parameters similar to the resolution of measurement equipment resulted in reasonable results for the SCD system. The resolution used resulted in an MC with approximately 550 million states.

To visualize our results we plot projections of the data for different concentrations of the chemicals involved. Specifically, these projections show the safety probability for entire range of sorbitol and fructose levels for certain values of the five other variables. Multiple

selections of the five other variables can be chosen to show a more comprehensive view of the data.

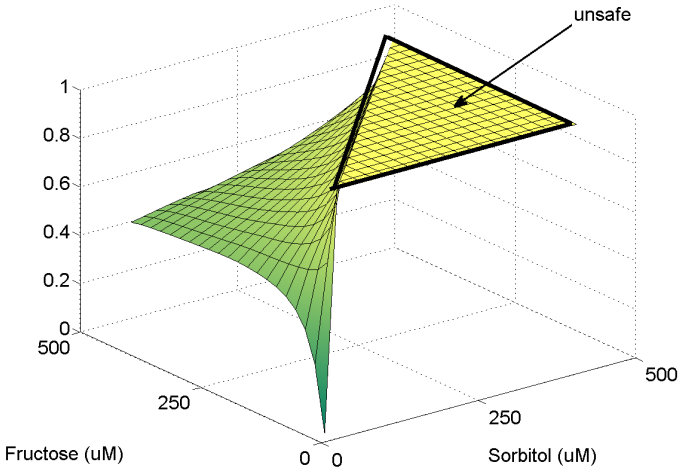
Figures 36 and 38 show projections of the value function for the safety and reachability results where $x_1 = 1.0$, $x_2 = 1.0$, $x_3 = 1.0$, $x_4 = 1.0$, and $x_5 = 0.1$. These figures show the safety or reachability analysis of the non-medicated SCD1 model (a), medicated SCD2 model (b) and medicated with delay SCD3 (c) model.

The differences between the safety verification results are shown in Figure 37 to highlight the differences between the analysis of the three models. Figure 37 (a) shows the difference between the SCD1 and SCD2 models. The difference between the value functions for these models is negligible for fructose values under 250 uM corresponding to the fact that the drug is not administered below 250 uM. Figure 37 (b) displays the difference that including the drug absorption and metabolization creates. The difference between SCD2 and SCD3 is especially large for situations where the concentrations of fructose and sorbitol are low.

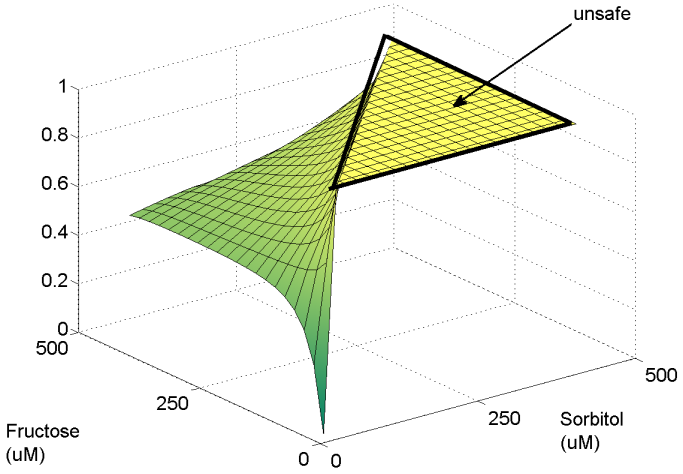
Figure 39 displays the differences between the calculated reachability values for SCD1, SCD2 (a) and SCD2, SCD3 (b). Both (a) and (b) show that the differences between the models are greater closer to the target set than the unsafe set. This implies that the effects of the drug are most influential to patients who are less likely to develop cataracts. This is most likely caused by the side effects of the drug that can adversely affect the patients fructose levels.

Analyzing the data generated by these experiments could possibly help predict sugar cataracts by demonstrating where the safest and most unsafe concentrations exist. Examining the differences between the various medication models could also possibly help further the understanding of how drugs are converted from prodrugs and metabolized. It could also give guidance for choosing the most effective or economical treatment to avoid cataract development. Furthermore, analysis can possibly guide doctors to better understand and predict why some drugs are more effective than others in different situations and better predict side effects for individual patients.

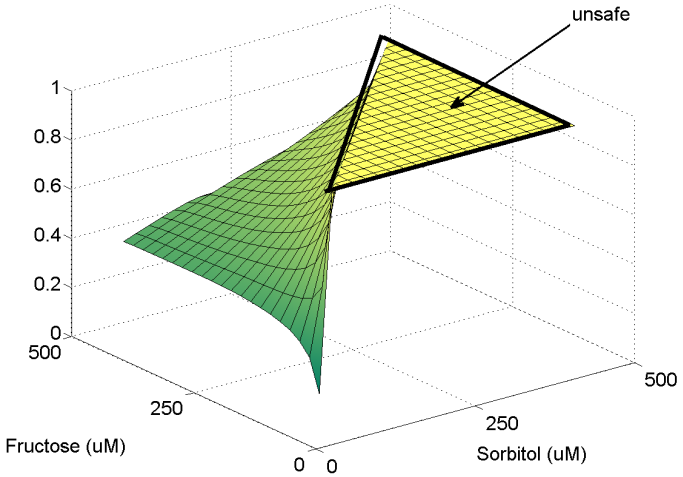
If the original dynamics of the model are changed by the user, the MDP must be regenerated with the new dynamics and the value iteration must be rerun. The computed value function from the original model can be used as the initial value function for the



(a) SCD1

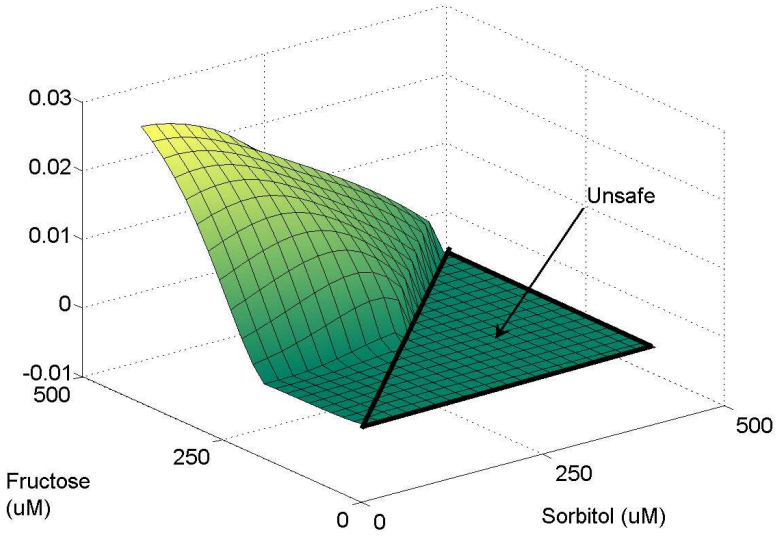


(b) SCD2

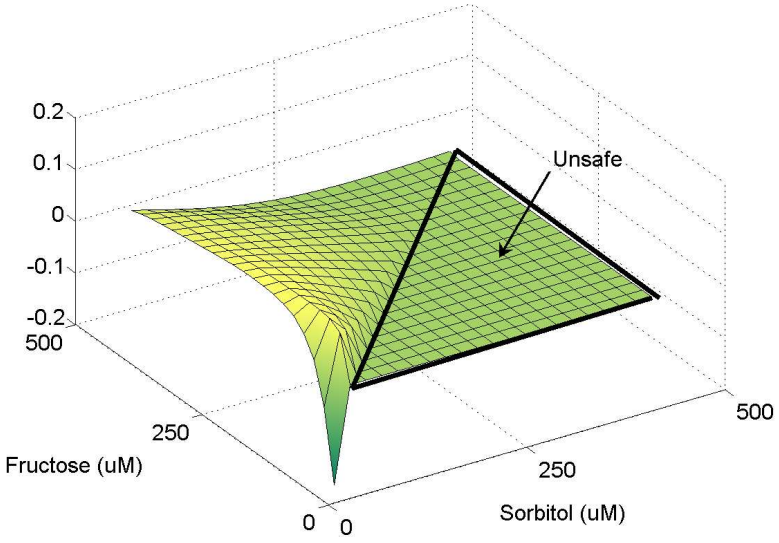


(c) SCD3

Figure 36: Safety results for SCD1, SCD2, and SCD3

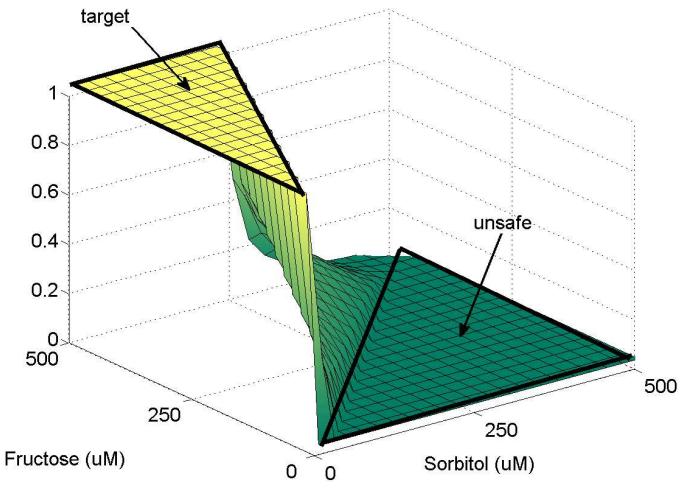


(a) SCD1 - SCD2

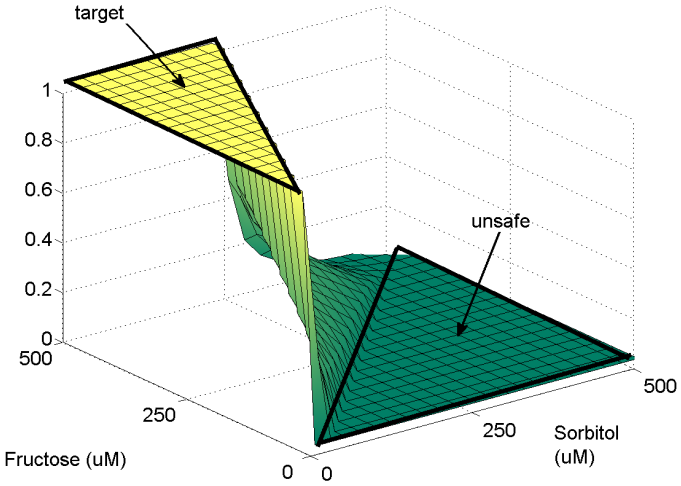


(b) SCD2 - SCD3

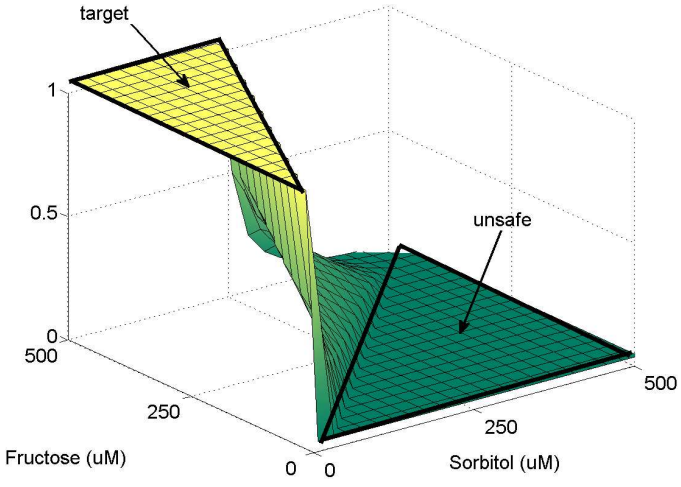
Figure 37: Differences between the safety results for the SCD models



(a) SCD1

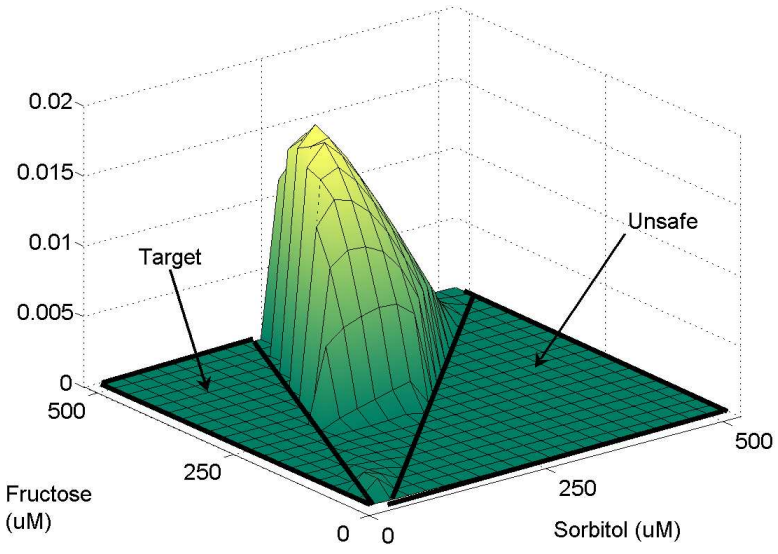


(b) SCD2

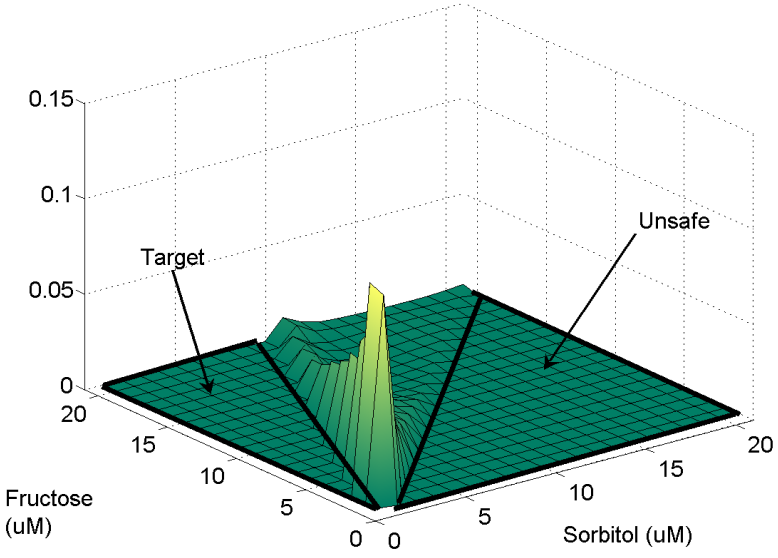


(c) SCD3

Figure 38: Reachability results for SCD1, SCD2, and SCD3



(a) SCD1 - SCD2



(b) SCD2 - SCD3

Figure 39: Differences between the reachability results for the SCD models

altered model to significantly improve the efficiency of the value iteration convergence. The amount of the improvement depends on the dynamics of the system and the changes that are made. The difference between the original model and the new model can be easily quantified using statistical methods by comparing the original value function and the newly computed result.

Biodiesel Processor

We use the exhaustive verification algorithm to verify the VTBD and CTBD models using parallel methods to improve the efficiency of the analysis. The value iteration algorithm is still guaranteed to converge in a parallel implementation as long as updated values are used periodically [19]. We use the range values for each variable presented in Table 5. Since the ranges for the variables are different, multiple individual resolutions must be considered (Table 14). The resolutions were chosen by decreasing each step size individually until no appreciable difference between the value functions at differing step sizes is observable. We then set $h = 1$ for scaling the entire system. These resolutions result in a state space consisting of almost 500 million states.

Table 14: Resolution for the VTBD model

Reactant	Resolution Scaling (M)
TG	0.125
DG	0.125
MG	0.125
E	0.5
M	0.5
Gl	0.25
T	10

To visualize our results we plot projections of the data for different concentrations of the chemicals involved. Our figures display the full ranges of monoglycerides (MG) and Esters (x_3, x_4) for under the following restrictions $x_1 = 0.00001$, $x_2 = 1.0$, $x_5 = 9.0$, $x_6 = 0.5$, and $x_7 = 70.0$. Figure 40 shows the projection of the reachability probability for the VTBD model, Figure 41 shows the projection of the reachability probability for the CTBD model, and Figure 42 shows the difference between the two figures. The target set is indicated in

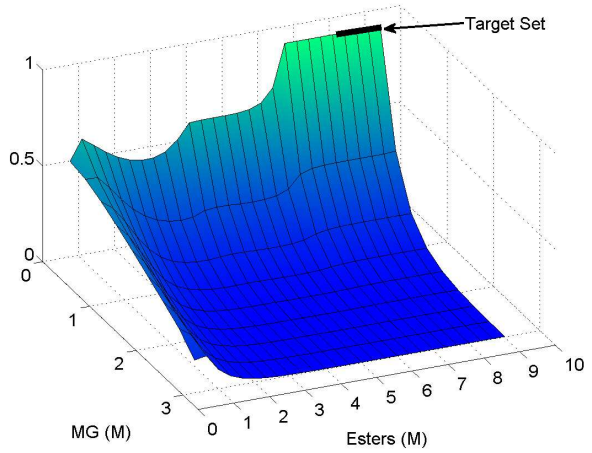


Figure 40: Value function for the VTBD reachability results

the figures.

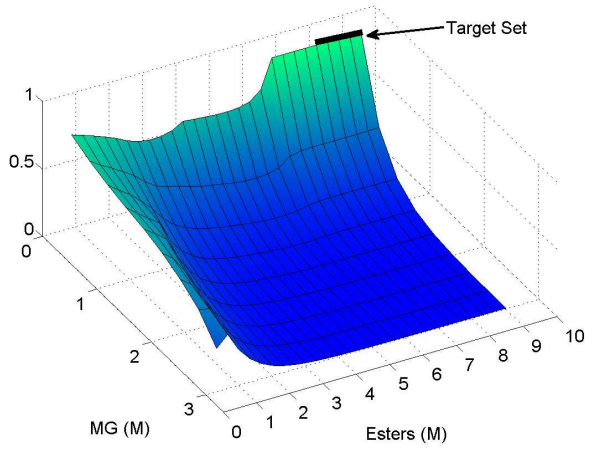


Figure 41: Value function for the CTBD reachability results

It can be seen in the figures that the temperature model significantly affects value function of the system. These results indicate that the temperature controller does not work effectively for this system because the probability of success for many of the states is fairly low. Further experiments can be performed to determine the ideal temperature to use the heater to maximize efficiency and minimize the use of the heater.

Summary

Advanced analysis methods such as exhaustive verification are useful because verification can be used to determine properties such as reachability or safety for a system. Exhaustive

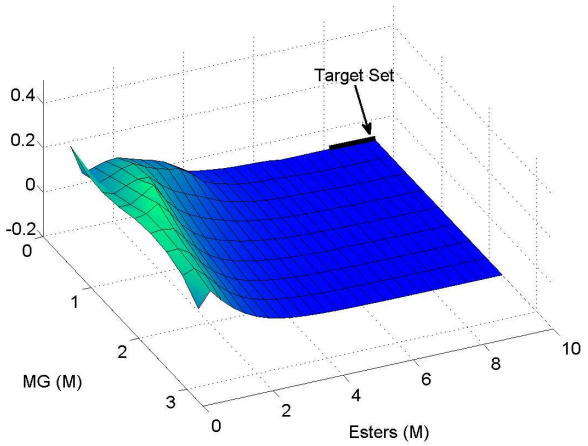


Figure 42: Difference between the value functions for the VTBD and CTBD models

verification of reachability properties for SHS aims at determining the probability that the system reaches a set of desirable or unsafe states. Reachability analysis is an important problem because it provides a formal framework to analyze complicated, realistic systems that operate in the presence of uncertainty and variability.

Developing sound computational methods for verification is challenging because of the interaction between the discrete and the continuous stochastic dynamics. These interactions cause discontinuities that are difficult to handle with many systems, so a framework is necessary to ensure these dynamics are handled properly. Further, computational restrictions of exhaustive verification can limit the size of the system that can be handled, so efficient, scalable methods are required.

In this chapter we present a dynamic programming verification method for SHS that utilizes Markov decision processes and a value iteration technique for analysis. We also present a parallel decomposition of the algorithm that can be used to scale the analysis method for significantly large and realistic systems. While the method is constrained by the curse of dimensionality, we show that it is useful for realistic systems by presenting experimental results for a navigation benchmark, a room heating benchmark, the sugar cataract development system, and the biodiesel system. We present the reachability or safety analysis results for each system as well as the performance results to show the methods' applicability to realistic systems.

CHAPTER VI

MONTE CARLO METHODS FOR SHS

Exhaustive verification can provide significant analysis information for Stochastic Hybrid System (SHS) models, but many realistic models are too large for exhaustive verification analysis. Therefore, it is important to develop scalable analysis methods that can determine reachability or safety properties. Monte Carlo methods can be used to determine reachability and safety probabilities for large SHS systems [114].

Monte Carlo methods for SHS must use accurate, efficient simulation methods to ensure the analysis results are reliable. If events critical to the system happen rarely, the accuracy and efficiency of the Monte Carlo analysis will be significantly diminished. Variance reduction methods can be used to increase accuracy and efficiency of Monte Carlo methods for these systems [93]. Variance reduction methods use information known about the system to refine simulations in predetermined regions. Variance reduction methods such as MultiLevel Splitting (MLS) must be carefully implemented for SHS to ensure boundary conditions are handled properly for improved efficiency. Further, large systems with rare events may still require prohibitively large numbers of simulations, so parallel methods may be necessary.

In this chapter we present a method for solving the reachability and safety problems for SHS using Monte Carlo methods. Further, we develop a variance reduction method using MLS for SHS. We demonstrate an effective method for choosing boundary placement and splitting policies for SHS to demystify the parameter choices for such systems. We also develop a parallel version of our methods to provide further efficiency enhancements along with scalability analysis using the glycolysis model.

We present experimental results to demonstrate the variance and efficiency gains for various MLS parameters using the glycolysis and Variable Temperature BioDiesel (VTBD) models. We next present a comparison of our exhaustive verification method with the Monte Carlo methods to show the correctness of both methods. We conclude this chapter with a comparison of traditional Monte Carlo methods with our variance reduction methods using MLS to demonstrate the performance and accuracy gains attainable.

Reachability Analysis Using Monte Carlo Methods

Consider a strong Markov process $\{s(t)\}$, we define two disjoint subsets U and T for the unsafe and target sets respectively. The stopping times $\tau_U = inf\{t > 0 : s(t) \in U\}$ and $\tau_T = inf\{t > 0 : s(t) \in T\}$ occur when the trajectory hits either the unsafe or target set. For the reachability problem we want to determine the probability $P_R = \mathbb{P}[\tau_T < \tau_U < \tau_{max}]$, or that $s(t)$ will hit the target set T without first hitting the unsafe set U on the time interval $(0, \tau_{max}), \tau_{max} < \infty$.

Monte Carlo methods estimate P_R by executing n independent simulations of the process $\{s(t)\}$. The number of trajectories that reach the set T before reaching the set U or time τ_{max} are divided by the total number of trajectories n to determine the reachability probability. This is given by $\widehat{P_R} = \frac{1}{n}\sum_{i=1}^{n} H_{R,i}$ where

$$H_R = \begin{cases} 1 & \text{if } \tau_T < \tau_U < \tau_{max} \\ 0 & \text{otherwise} \end{cases}$$

The formulation of the reachability problem can be modified to describe safety. For a safety problem, we are given a set of unsafe states and we want to compute the probability that the system execution from an arbitrary (safe) initial state will avoid the unsafe set. It is given by $P_S = \mathbb{P}[\tau_U < \tau_{max}]$.

Monte Carlo methods can also be used to estimate P_S. The number of runs that reach the set U before time τ_{max} are divided by the total number of runs n to determine the safety probability given by $\widehat{P_S} = \frac{1}{n}\sum_{i=1}^{n} H_{S,i}$ where

$$H_S = \begin{cases} 0 & \text{if } \tau_U < \tau_{max} \\ 1 & \text{otherwise} \end{cases}$$

The variance of the hitting probability for Monte Carlo methods for both reachability

or safety is given by

$$Var[\widehat{P}] = \frac{\sum_{i=1}^{n} \left(H_i - \widehat{P}\right)^2}{n}$$

If n is very small, then the estimate $\widehat{P}$ will have a large variance and may not be reliable. The only way to reduce the variance of the estimator using traditional Monte Carlo methods is to increase n [93, 127]. Rare events in a system can strongly increase the variance of the reachability or safety results of Monte Carlo methods. As an influential event becomes more rare, the error it can create increases dramatically, so variance reduction methods for rare events are necessary.

Rare Event Detection Using Multilevel Splitting

MLS is a variance reduction method for rare events that extends Monte Carlo methods by splitting individual trajectories of the Monte Carlo estimator in the region of the a rare event. This regional splitting reduces the variance of the estimator by increasing the density of the trajectories in the region near the rare event, but care must be taken to choose when and how the trajectories are split to guarantee reasonable efficiency improvements. We define the region of the state space where the rare event exists A as a subset of the state space.

Regions of the state space may include events that occur rarely but have a large influence on the system. We define a region of the state space where an influential event is reached with a probability of less than two percent as a rare event region A. We define MLS splitting levels which create proper supersets of the set A: $A \subset A_1 \subset A_2 \subset \ldots A_g$. When a simulated trajectory crosses from a larger set A_k into a smaller set A_{k-1}, the trajectory is split into j new trajectories which evolve using unique Wiener processes. An example MLS scenario is shown in Figure 43.

Trajectories are assigned importance values v_i to represent the amount of influence the trajectory has on the approximation. Initially $v_i = 1/n$ where n is the original number of trajectories. When a trajectory is split, the importance value is divided evenly between the split forks of the trajectory, and the total number of trajectories n_m is incremented

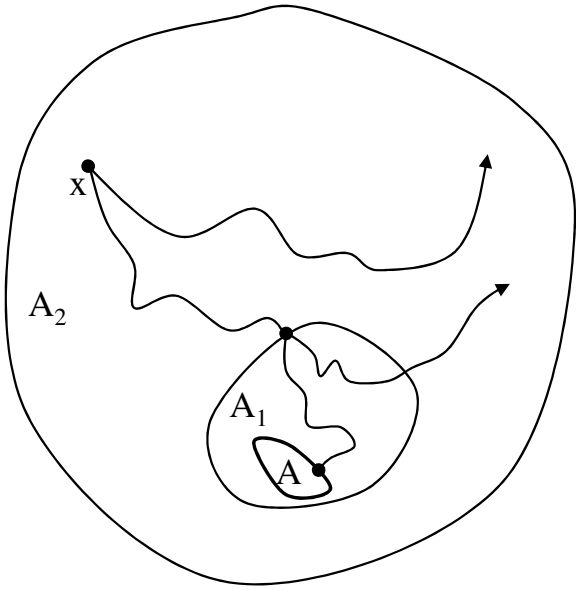


Figure 43: An example MLS scenario

$n_m+ = j - 1$. Multiple splitting policies can be used to split trajectories differently at different levels according to the variance reduction desired.

The variance of the Monte Carlo estimator is reduced by increasing the number of samples n to n_m for a region of the state space around A. An artificial drift is created toward the region A by the reinforcement of trajectories through splitting. The variance reduction is unbiased despite the fact that the trajectories are not completely independent [46]. Further, the variance reduction is accomplished with a significantly improved efficiency compared to traditional Monte Carlo methods [93].

Reachability and Safety with Rare Events

Rare events may occur in any region of the state space, but they are most often found as part of the unsafe set U, so we assume that the rare set is given by $A \subseteq U$. We define rare event problems for the safety and reachability problems where the rare event is found in the unsafe set.

Safety Problem MLS methods can be adapted for safety analysis to reduce the variance of the Monte Carlo estimator. The safety probability for MLS is determined by $\widehat{P_S} = \sum_{i=1}^{n_m} H_{S,i} v_i$. Splitting in the region near A will increase the total number of trajectories n_m

and change the influence of trajectories that are split v_i.

An example of where the rare event may be found in the unsafe region is an air traffic control problem where the unsafe region defines a collision. A collision is a rare event for most simulation trajectories, and it is very important to have a high confidence in the estimate of the probability of a collision. A graphical example of this is shown in Figure 44.

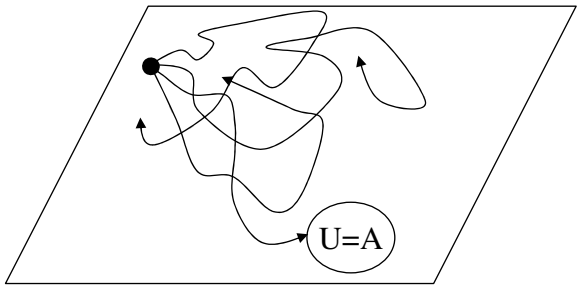


Figure 44: MLS for safety analysis

Reachability Problem MLS methods can be adapted for reachability analysis to reduce the variance of the Monte Carlo estimator. The reachability probability for MLS is determined by $\widehat{P_R} = \sum_{i=1}^{n_m} H_{R,i} v_i$. Splitting in the region near A will increase the total number of trajectories n_m and change the influence of trajectories that are split v_i.

An example of where the rare event may be found in the unsafe region for the reachability problem is a drug administration model where it is rare and unsafe for a patient to die, and where the goal or target of the policy is to reach a healthy state. In this situation it is very important to understand the probability that the patient successfully recovers and avoids death. A graphical example of this is shown in Figure 45.

Multilevel Splitting for SHS

MLS with discrete mode changes requires further care to ensure the problem is solved accurately and efficiently. To extend MLS for SHS the standard notion of a Markov process is extended to include discrete modes $q(t)$: $\{s(t)\} = \{x(t), q(t)\}$. We can redefine the stopping times $\tau_U = inf\{t > 0 : s(t) \in U\}$ and $\tau_T = inf\{t > 0 : s(t) \in T\}$ to extend the notions of reachability and safety to the hybrid case.

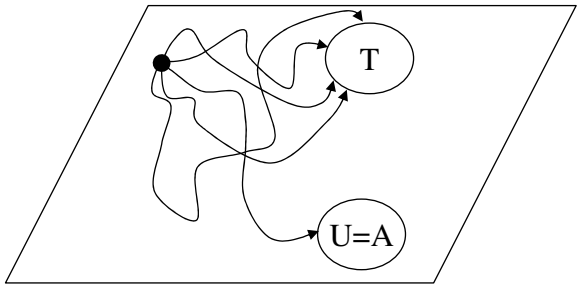


Figure 45: MLS for reachability analysis

The discrete transitions found in SHS can cause discontinuities that require special care in the presence of splitting boundary crossings. Figure 46 shows an example SHS where the trajectory crosses a splitting and discrete boundary simultaneously. The trajectory starts at state $s_0 = (q_1, x_0)$, and evolves until it reaches the boundary for A_2 or the guards for a discrete transition are satisfied. In this example both the discrete transition is fired and the splitting level is crossed simultaneously, and the reset of the discrete transition updates the state of the trajectory to $s = (q_2, x_t)$.

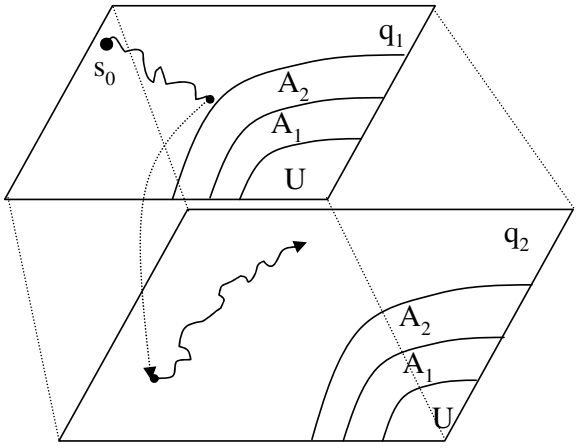


Figure 46: Example MLS problem in a hybrid state space

Because the new state is not in the splitting region A_2, splitting the trajectory before applying the reset may not necessarily reduce the variance, and could decrease the efficiency, so it should be avoided. This problem is further exacerbated if the splitting coefficient j

is large. Therefore, care must be taken to ensure that discrete transitions are fired before testing splitting boundaries.

Another situation that must be handled is where the trajectory begins outside a splitting region, and the reset causes the trajectory to jump into a splitting region A_i. In this case, it is important to split the trajectory if it has not been previously split to maintain the greatest variance reduction. It is possible that the trajectory will jump into a region such as A_1 before it has entered the superset A_2. In this case, the splitting coefficient j must be chosen to ensure the variance is effectively reduced while the efficiency is not unnecessarily decreased. Our algorithm tests for these cases to ensure that they are handled appropriately.

Further, the use of accurate simulation methods including detection and handling of boundaries is important because MLS and Monte Carlo methods require highly accurate trajectories to ensure appropriate estimates. If low order methods or large step sizes are used, the Monte Carlo methods may not provide a reliable result.

Unfortunately, MLS cannot reduce the variance of an estimator for systems where rarity is created by events that cannot be decomposed into a series of less rare events. These systems can also be identified through sample trajectory testing.

SHS Multilevel Splitting Algorithm

We create a depth-first implementation of the SHS MLS algorithm that handles the discrete splitting challenges presented earlier. Splitting boundary crossings are tested after every step, and if the trajectory crosses a splitting boundary at time t, the state $s_t = (q_t, x_t)$ is saved for future splitting. After the original trajectory is complete, the most recently saved state is reloaded and the simulation is continued with a new Wiener process. Multiple splits may occur, so all split trajectories must be completed before a new Monte Carlo trajectory is begun. This depth-first approach to simulating split trajectories requires a small amount of memory and computational overhead. The split trajectories evolve according to unique Wiener processes w_j thus reducing the variance of the overall estimate.

We also consider the handling of transitions between the modes because a discrete transition may cause the trajectory to cross one or more splitting boundaries, i.e. $s_t \notin A_2$ *and* $s_{t+\Delta t} \in A_1$. Therefore we test for this and split the trajectory in the new state as

many times at it would have been split had it crossed through all the levels separately. This ensures that the variance reduction is preserved.

We extend our previous simulation algorithms to implement the MLS technique for SHS. Each step of the extended simulation method includes testing and forking for the splitting boundary crossings. The boundary crossing conditions are tested after the state is fully updated and any discrete transitions are completed to avoid the potentially inefficient situation where the discrete transition and splitting level conditions are both satisfied, but the reset moves the state to a region away from A. $StartNextSplitTrajectory$ keeps a list of the split trajectories and conditions when the trajectories were split and starts the most recently split trajectory in the **while** loop. If no split trajectories exist, the function exits the **while** loop and starts the next new trajectory in the **for** loop. The pseudocode for the algorithm is given below for the fourth type of rare event scenario where A_k is the location of the nearest MLS boundary. The pseudocode can be easily manipulated to handle the other three types of rare event scenarios by adjusting the control conditions for the **while** loop and **if** check within the **while** loop.

Algorithm 6.1: MLSforSHS(τ_{max})

for $j = 1; j < n; j++$

$t = 0$

$ResetInitialConditions()$

$influence_j = \frac{1}{n}$

while $X_t \notin U$ **and** $X_t \notin T$ **and** $t < \tau_{max}$

do

$ATHMMstep(X_t)$

if $X_t \in A_k$

then $ForkTrajectory(j), split(influence_j)$

if $X_t \in U$

then $unsafecount+ = influence_j, StartNextSplitTrajectory$

if $X_t \in T$

then $targetcount+ = influence_j, StartNextSplitTrajectory$

if $t > \tau_{max}$

then $StartNextSplitTrajectory$

return $(\frac{targetcount}{n}, \frac{unsafecount}{n})$

MLS Parameter Selection

MLS has the potential to significantly reduce the variance and improve the efficiency of the estimator; however, set placement and splitting policies must be appropriately chosen to ensure the method performs well. If the A_k sets are chosen to be too close to A, not enough splitting will occur, and the variance reduction will be small (although the efficiency will be high). If the sets are too far away, too much splitting may occur, and the efficiency will be adversely affected without significant variance reduction. Splitting more trajectories at each set boundary has the potential to further reduce variance, but if the boundaries are placed improperly, the efficiency can be significantly decreased without a significant decrease in variance.

There is no universally optimal method for choosing the placement of the sets A_k for a multidimensional system; however, it has been determined that they should be chosen to cause splitting of the trajectory in regions that are most likely to lead to the rare set A for

optimal efficiency [93]. Because this choice is crucial to the efficiency of MLS, we introduce a general method for determining the best locations to place the boundaries.

First, it must be determined which states are most likely to transition to the set A. To do this we use Monte Carlo methods with initial conditions in the region near A to determine the safety probability of each location reaching the rare set A. Using more starting locations provides more accurate information but takes more computational time, so a trade off must be found.

We can use the safety probabilities in the region near A to determine where the sets are best placed by including the regions that are most likely to lead to the set A. The region of highest probability (typically $> .9$) of leading to A is chosen to be A_1. Monte Carlo simulations can then be used to determine the regional probabilities of transitioning to the new region A_1, and A_2 can be defined by the states that have highest probability of leading to A_1. This method can be used to recursively define all sets A_k.

The number of levels to use k must be determined to create an efficient implementation of the MLS algorithm. Using more levels has the potential to reduce the variance further, but also decreases the efficiency (sometimes significantly). Therefore, a trade off must be found to choose the number of levels to use. Typically a small number of levels (< 5) reduces the variance while maintaining sufficient efficiency.

The splitting coefficient j is another important parameter that has implications on the performance and accuracy of MLS. If more trajectories are forked, the overall efficiency is decreased, but the variance may also be significantly decreased.

If the A_k sets and splitting policy are chosen well, the efficiency gains over traditional Monte Carlo simulation can be significant. Efficiency can also be further improved by terminating trajectories if they stray far from the rare event region A, target set T, or unsafe set U. This ensures that trajectories that do not influence the outcome of the system do not reduce the efficiency of the approximation; however, care must be taken to ensure trajectories are not prematurely terminated if they may eventually affect the system outcome.

MLS Accuracy and Efficiency

MLS methods can reduce the variance of Monte Carlo methods, and thus increase the accuracy of the approximation by increasing the number of simulations in a certain region. The specific amount of variance reduction is dependent upon the system dynamics, the placement of the sets A_i, and the splitting policy, but MLS has the potential to reduce the variance by at least an order of magnitude [93]. The hybrid state space increases the difficulty of determining the optimal boundary placement and splitting coefficient j, but most non-optimal solutions still provide good variance reduction.

If the variance reduction is necessary for only a very small region of the state space, the splitting regions can be small to ensure a large reduction in variance and efficiency increase. If the region of interest is larger, then the levels can be arranged to encompass a larger region, but the variance reduction may not be as significant and the efficiency may not improve as much.

The efficiency of the estimator $\widehat{P}$ is dictated by the set placement, splitting policy, and dynamics of the model. We define the efficiency as $Eff\left[\widehat{P}\right] = \frac{1}{Var\left[\widehat{P}\right]C(\widehat{P})}$ where $C\left(\widehat{P}\right)$ is the expected execution time to compute the estimator [93]. The efficiency can be increased by decreasing the variance and/or decreasing the computation time.

Simulating more trajectories decreases the efficiency of the estimator by increasing the execution time C, so it is important to ensure that the trajectories are split in regions of interest, so the variance is ultimately reduced to improve efficiency. MLS decreases C quickly compared to traditional Monte Carlo methods by partially reusing previously computed paths and therefore reducing the cost C to achieve the same variance reduction Var for a limited region of the state space. Knowledge of the dynamics in the regions of interest is important to determine the most effective placement of the boundaries to ensure efficient and accurate results. For SHS, it is important that the most up-to-date state information is used to determine if switching boundaries are crossed to ensure that wasteful splitting does not increase C.

Parallelization

The number of required simulations may still be quite large even when using variance reduction methods, so parallel Monte Carlo methods using MLS methods may be necessary. There are no dependencies between the individual trajectories, so the algorithm can be parallelized by running multiple trajectories concurrently on multiple processors. After all trajectories are complete on all processors, the results can be compiled and reported. Because the collection of the results is the only overhead necessary, the speedup is nearly linear, so parallelization is quite effective at improving the efficiency. This type of parallelization has been used previously with Monte Carlo methods [137], and care must be taken to ensure that the random number generators used to generate the Wiener processes do not introduce bias.

Experimental Results

We present the experimental results of our Monte Carlo methods using MLS to demonstrate their usability and performance enhancements. We begin by presenting methods for choosing the splitting policy and boundary placement. We use the glycolysis and VTBD models to demonstrate the implications of these parameter selections because they are large, complex, realistic systems. We show that small adjustments in the splitting policy or boundary placement can significantly effect the variance and efficiency of the method. Results from a comparison of traditional Monte Carlo methods and Monte Carlo methods using MLS are presented as well. We present this comparison to motivate the use of MLS and provide perspective for the performance and accuracy gains.

We next compare the Monte Carlo reachability results with reachability results calculated using the exhaustive verification method. We present a model of a double integrator which is a small, simple model that can be easily verified. We compare the results of the double integrator analysis for the entire state space to show the robustness of the algorithm. We next present a comparison of the Monte Carlo reachability analysis and exhaustive verification analysis for the VTBD model. The Monte Carlo results are only presented for a subset of the state space because the state space is so large. We also present parallel

performance results to demonstrate the performance potential of Monte Carlo methods.

Choosing Parameters for MLS

Glycolysis Model Glycolysis is understood as a global process, but the subtleties of the interrelations between the individual reactions are still not fully understood. Simulating the model under various conditions can reveal intricacies of the model, and analysis of the model with Monte Carlo methods can further enhance the understanding of both the global and local behavior of the model. These understandings can help to expose inaccuracies in the model that correlate to misunderstandings in the modeled system.

Glycolysis is a process that turns glucose into energy in a cell, so the supply of glucose is crucial to the function of the system. If the amount of glucose drops below a certain level, a cascade of problems is set off ultimately leading to insufficient energy production and potentially cell death. Therefore, we define an unsafe condition for the system when $Glc_x \leq 2.75$, and we examine the safety probability of the system. The probability of the system reaching the unsafe state is small, so the rare event region $A \subseteq U$. We perform analysis of the system using initial conditions from Table 7, and we generate safety probabilities for the system.

To evaluate the variance and efficiency of Monte Carlo methods, we tested the outcomes of the safety probability for the glycolysis system using various numbers of iterations n. The results of this analysis can be seen in Figure 47. It can be seen from the figure that increasing the number of iterations n decreases the efficiency and the variance, but the efficiency decreases much faster than the variance. This motivates the need for variance reduction methods to improve efficiency.

We first demonstrate the importance of choosing appropriate splitting levels by examining MLS methods with various placements of the splitting boundaries. We examine the safety problem for the glycolysis model where $Glc_x < 2.75$ is the unsafe region. The starting value for $Glc_x = 4.25$ usually causes the system to avoid the unsafe region, but it is reached rarely, so it is a rare event scenario of type 1.

We use three splitting levels ($L1$, $L2$, and $L3$) as shown in Figure 48. We used three different placements of the splitting boundaries where placement A uses a wide spacing

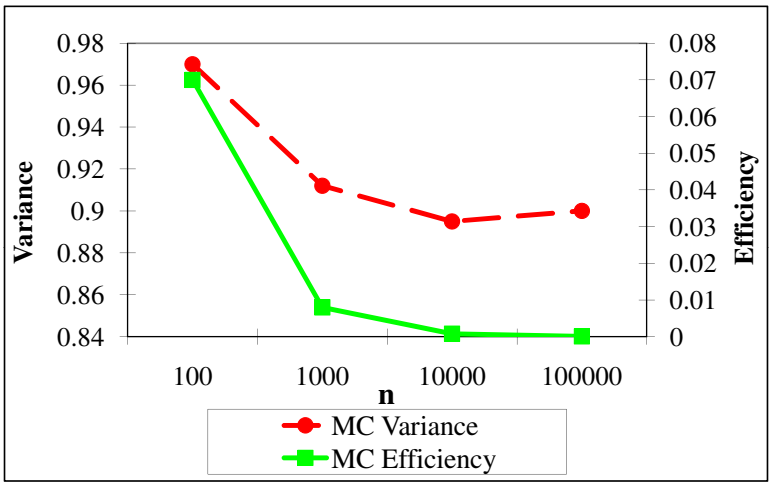


Figure 47: Monte Carlo results for the glycolysis model

near the unsafe region, placement B uses a medium spacing, and placement C uses a tight spacing.

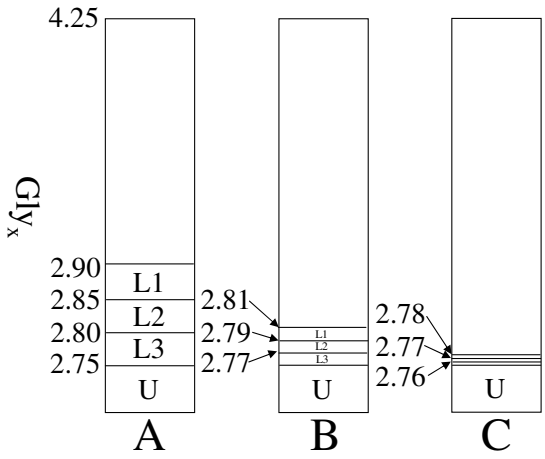


Figure 48: Boundary placement scenarios for the glycolysis model

We demonstrate the importance of choosing appropriate MLS splitting policies by examining three different example splitting policies. We consider three splitting levels for the glycolysis model ($L1$, $L2$, and $L3$) and we use three different splitting policies where policy I splits all trajectories two times at each level, policy II splits all trajectories four times at each level, and policy III splits the trajectory in two at $L1$, in four at $L2$, and six at $L3$. Examples of these splitting methods are shown in Figure 49.

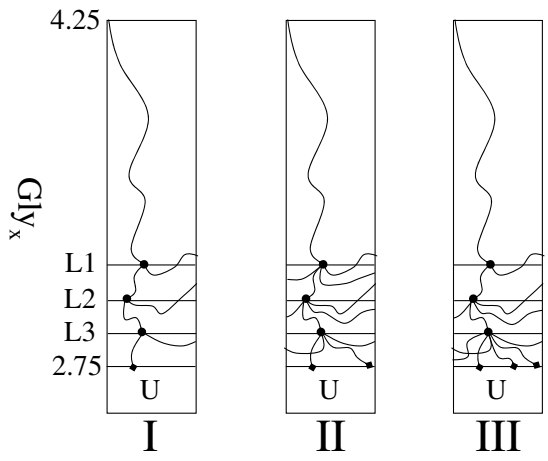


Figure 49: Splitting policies for the glycolysis model

We compared the three boundary placement schemes and three splitting policies using 1000 initial MLS trajectories. In Figure 50 we show the variance results for all nine possible combinations of methods. The variance is significantly reduced when using MLS in comparison to traditional Monte Carlo methods; however, the choice of splitting policy and boundary placement is important. It can be seen that the variance is reduced the most by using the boundary placement scheme B with splitting policy II.

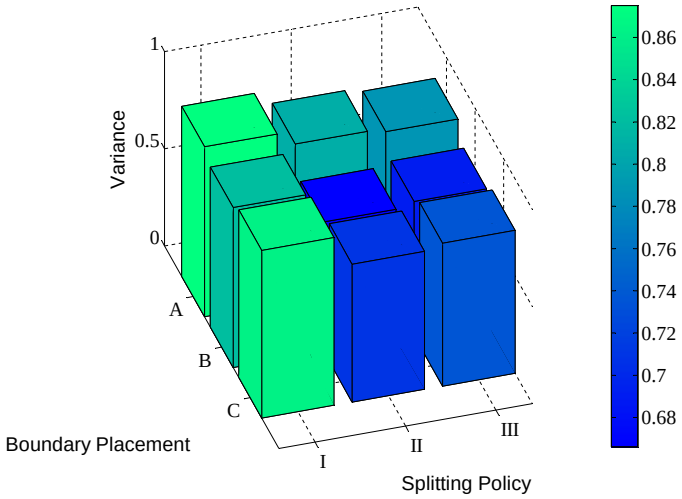


Figure 50: Comparison of variance for the MLS methods for the glycolysis model

In Figure 51 we show the efficiency results for all nine possible combinations of methods.

It can be seen that the efficiency is largest when using the boundary placement scheme C with splitting policy I; however, this combination has a large variance. Compromises between efficiency and variance can be found depending on whether variance reduction or efficiency improvement are higher priority. Any of these methods work adequately to reduce the variance and improve the efficiency when compared to traditional Monte Carlo methods. Tuning to the methods holds potential for further variance reduction and efficiency gains.

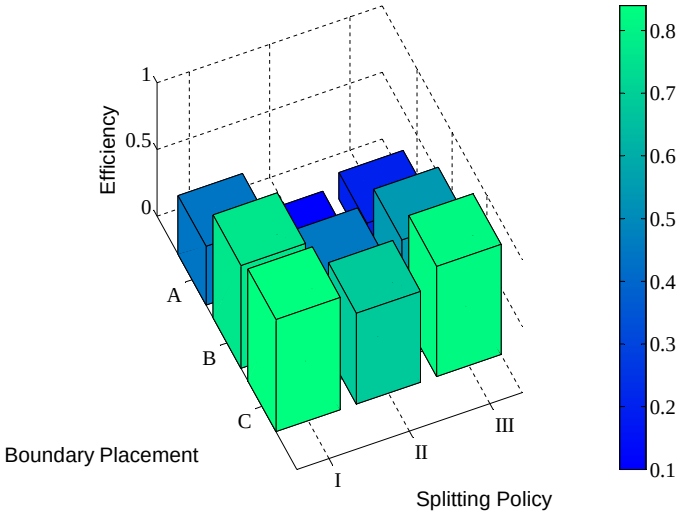


Figure 51: Comparison of efficiency for the MLS methods for the glycolysis model

Biodiesel Model We examine the variance and efficiency of Monte Carlo methods for the VTBD model to better understand performance and accuracy of Monte Carlo methods and compare with our variance reduction methods. We analyze the VTBD model for reachability to determine whether biodiesel is successfully produced. The unsafe conditions where methanol is depleted rarely happens, but when it does, the system will be detrimentally affected, so we use reachability analysis with MLS methods to analyze this system.

We tested simple Monte Carlo simulations of the model using various numbers of iterations n to demonstrate the variance and efficiency of the methods. The results of this analysis can be seen in Figure 52. It can be seen from the figure that increasing the number of iterations n decreases the efficiency and the variance, but the efficiency is decreased significantly faster than the variance. This motivates the need for variance reduction methods to improve efficiency.

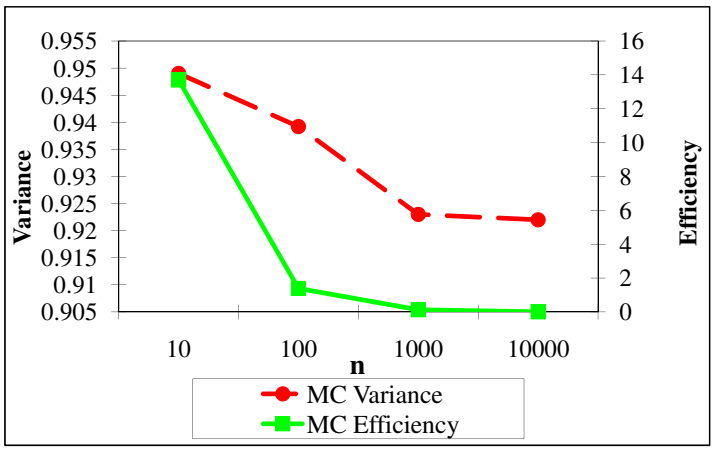


Figure 52: Monte Carlo results for the VTBD model

We demonstrate the importance of choosing appropriate splitting levels by examining MLS methods with various placements of the splitting boundaries. We examine the reachability problem for the VTBD model which is a rare event scenario type 2 because the unsafe set is hit rarely. We use three splitting levels ($L1$, $L2$, and $L3$) as shown in Figure 53. We used three different placements of the splitting boundaries where placement A uses a wide spacing near the unsafe region, placement B uses a medium spacing, and placement C uses a tight spacing.

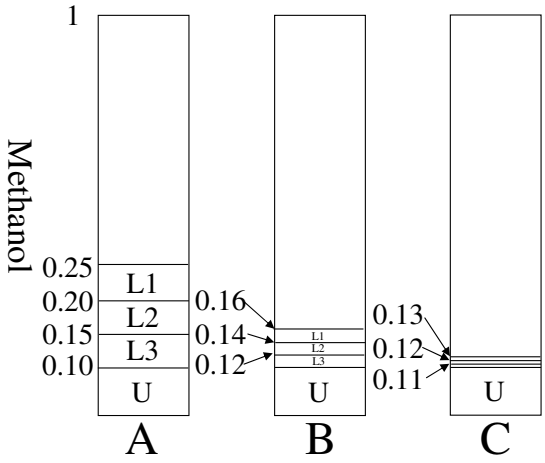


Figure 53: Boundary placement scenarios for the VTBD model

We demonstrate the importance of choosing appropriate MLS splitting policies by examining three different example splitting policies. We consider three splitting levels ($L1$, $L2$, and $L3$) and we use three different splitting policies where policy I splits all trajectories

two times at each level, policy II splits all trajectories four times at each level, and policy III splits the trajectory in two at $L1$, in four at $L2$, and six at $L3$. Examples of these splitting methods are shown in Figure 54.

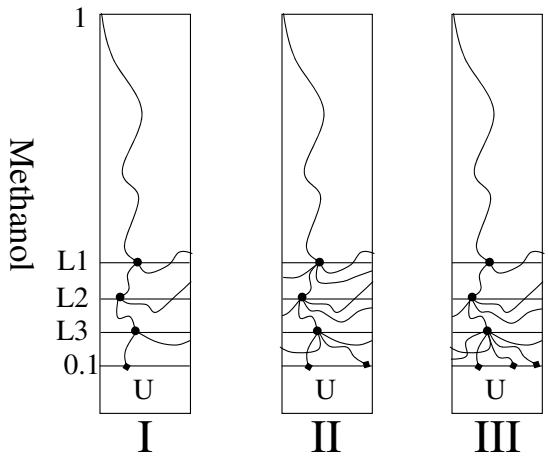


Figure 54: Splitting policies for the VTBD model

We compared the three boundary placement schemes and three splitting policies using 1000 initial MLS trajectories. Figure 55 shows the variance results for all nine possible combinations of methods. All the methods we tested reduced the variance far more than traditional Monte Carlo methods with $10,000$ trajectories, but some reduced the variance more than others. It can be seen that the variance is reduced the most by using the boundary placement scheme C with splitting policy II.

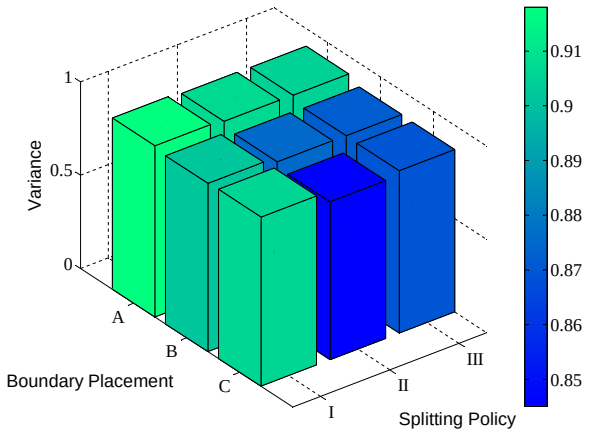


Figure 55: Comparison of the variance for the MLS methods for the VTBD model

In Figure 56 we show the efficiency results for all nine methods. The efficiency for traditional Monte Carlo methods using $n = 1000$ and $n = 10,000$ are $Eff = .14$ and $Eff = .014$ respectively. While the MLS methods do not achieve the same efficiency as traditional Monte Carlo methods with the same number of iterations n, the efficiency is close, and the variance reduction is significant. It can be seen that the efficiency is largest when using the boundary placement scheme C with splitting policy I; however, this combination does not maximally reduce the variance. However, scheme C with splitting policy II produces the best variance reduction of our tests as well as good efficiency. Further refinements could be made to the methods to further enhance accuracy and efficiency, but these results show that analysis such as we present is sufficient to distinguish appropriate methods and significant gains over traditional Monte Carlo methods.

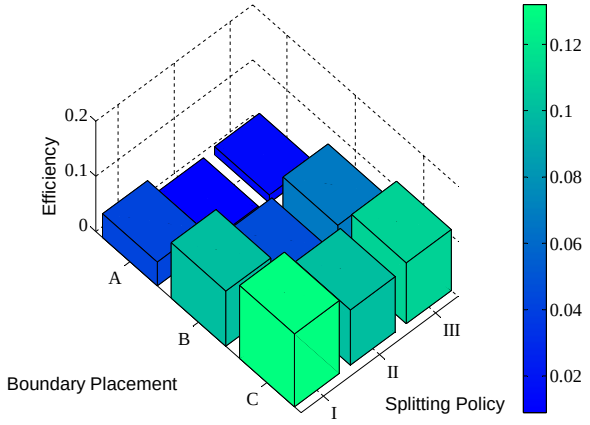


Figure 56: Comparison of the efficiency for the MLS methods for the VTBD model

Comparison with the Exhaustive Verification Method

To ensure that our Monte Carlo MLS analysis method is valid, we compare it with the results of the exhaustive verification method presented in the previous chapter.

Double Integrator The first SHS model we consider for comparison is the two-dimensional double integrator benchmark from [82]. The double integrator is a plant that is controlled by discrete actions. The double integrator model may used as a simple example for satellite control by modeling the relation between the angular position and velocity and the reaction jets. The differential equations that model the dynamics of the system are

given by

$$\begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix} = \left(\begin{bmatrix} x_2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(t) \right) dt + \begin{bmatrix} .1 \\ .1 \end{bmatrix} dw$$

where $r(t)$ can take values -1 or 1 depending on the state of the system. The boundaries of the state space are defined by $-\nu_1 < x_1 < \nu_1$ and $-\nu_2 < x_2 < \nu_2$. The goal is to drive the system to a steady state within the boundaries.

The analysis of the double integrator requires safety analysis, and does not have any rare events to consider. The unsafe region for this system is defined around the outside of the state space. We use this model because it is simple, small and can be easily exhaustively verified. It is also small enough that Monte Carlo analysis can be performed on it at the same resolution as the verification for complete comparison. Figure 57 shows the verification analysis results using our exhaustive verification technique. Figure 58 shows the results of the Monte Carlo analysis on the same system.

The Monte Carlo results were generated by performing the Monte Carlo analysis starting a trajectory at every grid point used by the exhaustive verification algorithm. It can be seen that the results are very similar, which demonstrates the correctness of both methods. The differences between the results can be explained by the different Wiener processes generated by the two methods. The analysis using the verification is several orders of magnitude faster because it is much more efficient at exhaustively analyzing a system.

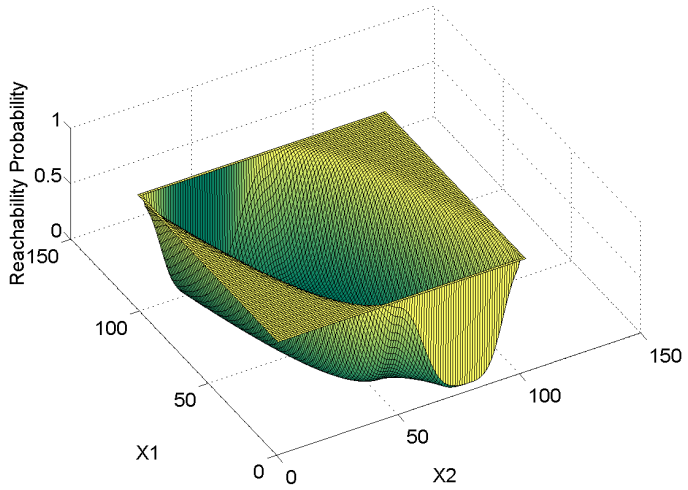


Figure 57: Dynamic programming verification results for the double integrator

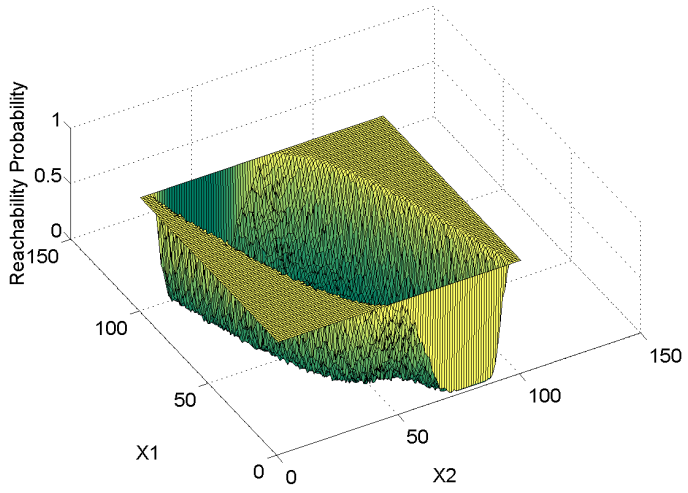


Figure 58: Monte Carlo analysis results for the double integrator

Biodiesel Model We compare the exhaustive verification results and parallel Monte Carlo results for the VTBD model to demonstrate the correctness of both approaches. Because the VTBD model is quite large, we consider only the portion of the state space. We use traditional Monte Carlo analysis, and we only incorporate MLS methods if a rare event is impactful on the outcome of the analysis for the specific initial conditions. Figure 40 shows the dynamic programming verification results, Figure 59 shows the Monte Carlo results, and Figure 60 shows the difference between the methods. The analysis shows a strong similarity between the results of the two methods. The differences between the results can be explained by the different Wiener processes used for the two methods.

Scalability of Parallel Methods

Glycolysis Model We performed experiments to test the parallel scalability of our algorithm using the glycolysis model with MLS. We found that the parallel MLS algorithm took virtually the same amount of time regardless of the number of processors used, as seen in Table 15. The slight super linear speedup can be explained by the variability of the noise of the system. Noise causes the splitting policies to use different amounts of time because of the varying times the splitting and boundary hitting happen.

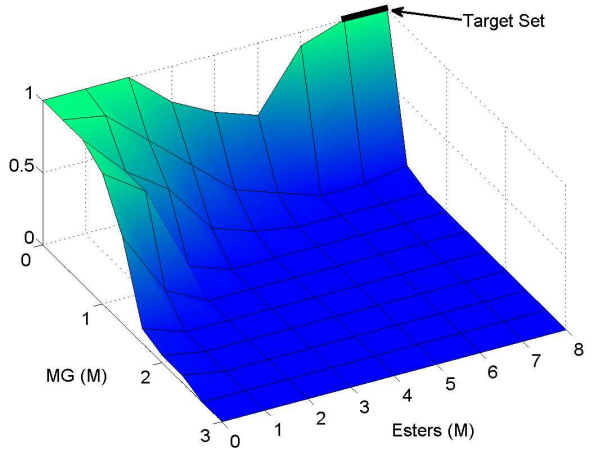


Figure 59: Monte Carlo analvsis results for the VTBD model

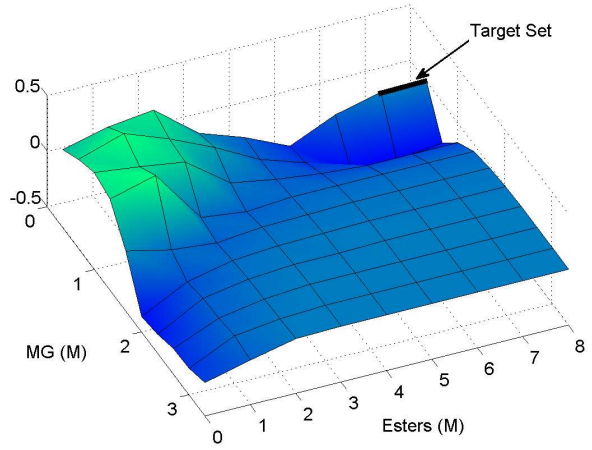


Figure 60: Difference between Monte Carlo and dynamic programming results

Summary

In this chapter we develop Monte Carlo methods for SHS along with variance reduction methods for rare events. These methods are useful for determining reachability or safety probabilities for systems that are too large to exhaustively verify using our dynamic programming method. In this chapter we formalize the reachability and safety problem and show how it could be solved using Monte Carlo methods. We also define efficiency as a performance metric and variance as an accuracy metric to quantify the improvements of our methods.

After describing the Monte Carlo and MLS techniques for SHS, we provide a description of the appropriate methods for choosing parameters for both methods. We demonstrate the

Table 15: Parallel performance results for the glycolysis model

Processors	Time to Execute
16	8.6
8	8.5
4	8.2
2	8.3
1	8.5

accuracy of our methods by presenting experimental results to demonstrate the performance and accuracy gains for MLS using the glycolysis and VTBD models. We also demonstrate the performance gains of using MLS over using traditional Monte Carlo methods.

In this chapter we compare the experimental results of the Monte Carlo analysis with those generated using our exhaustive verification method to demonstrate the correctness of both approaches. We use a simple double integrator model to demonstrate the results for the entire system and the VTBD model to demonstrate the results for a realistic model. We feel that these results demonstrate the accuracy and viability of these methods as realistic techniques for analyzing complex SHS.

CHAPTER VII

CONCLUSIONS

In this dissertation we develop a Stochastic Hybrid System (SHS) modeling methodology for biochemical systems as well as advanced simulation methods, verification methods, and Monte Carlo-based reachability analysis methods. We present several example systems that demonstrate the various benefits of our modeling methodology as well as our analysis methods.

Summary of Contributions

Our contributions focus on computation challenges in biochemical system modeling, simulation methods for SHS, verification methods for SHS, as well as Monte Carlo methods using variance reduction.

Biochemical Modeling using SHS We present a biochemical systems modeling framework for SHS that accurately models fast and slow dynamics in a stochastic framework [124]. We develop several case studies to demonstrate the modeling methodology. We create a SCD model with multiple types of drug treatments [124]. We develop a biodiesel production system model with temperature control and glycerol separation modeling [119]. We create a glycolysis model that is large and robust [125]. Finally, we develop a water/electrolyte balance system model that uses reflecting boundaries to describe physical limitations of the system [123].

SHS Simulation We develop improved SHS simulation algorithms that are utilize high-order simulation methods. We implement first and second-order Taylor-based simulation methods for SDEs [122]. We also implement improved methods for approximating absorbing and reflecting boundaries [122, 123]. We combine the Milstein method simulation algorithm with both improved absorbing and reflecting boundary methods to create a comprehensive weak order $\gamma = 1.0$ SHS simulation method [123]. We also develop and implement an adaptive time stepping algorithm for SHS that improves the accuracy and efficiency of the methods.

Computational Methods for Verification of SHS We present an exhaustive verification method for SHS based on dynamic programming. We develop a parallel version of the verification algorithm along with partitioning methods [120]. We use benchmarks and our case studies to demonstrate the effectiveness of the method for realistic systems. We present the reachability analysis of the room heater and navigation benchmarks [85]. We analyze reachability properties of the sugar cataract development system and consider the effects of various types of medication administration on the system [124]. We also analyze reachability properties of the variable temperature biodiesel model and the constant temperature biodiesel model, and we compare the results [119].

Monte Carlo Methods for SHS with Variance Reduction We develop an implementation of Monte Carlo-based reachability analysis methods for SHS using our advanced simulation methods [125]. We create MLS methods for SHS along with a method for choosing MultiLevel Splitting (MLS) parameters. We present examples of the MLS methods along with the efficiency and accuracy improvement results [126]. We develop and analyze a parallel version of the Monte Carlo methods as well. We also present a comparison of the MLS methods with our verification methods to demonstrate the correctness of both methods.

Future Directions

There are several directions that this research can take in the future. We present our ideas for further pursuit in the areas of modeling, simulation, verification, and variance reduction.

While there are many models of biochemical systems that currently exist, there are many more influential models that can be developed. Modeling traditionally lies in an interdisciplinary region where creation of a model requires expertise in not only the biochemical aspects of the system but also in the modeling and analysis paradigms. While our work in this dissertation aims at bridging the gap, further effort can be made to make modeling and analysis tools more usable for domain experts. Developing more examples will help, but additions such as plug-and-play tools or visual modeling methods are required to make these modeling and analysis tools more usable.

More complex, robust models can also be developed that will improve the general reliability of our modeling and analysis methods. Future models of complex biochemical phenomena should be developed with the collaboration of domain experts to not only demonstrate the usability of our methods, but also identify weaknesses and unanticipated challenges with the modeling and analysis tools.

While our improved simulation methods demonstrate an improvement over previous methods, further refinement can be achieved. Much development can be performed in the areas of adaptive time stepping to determine better algorithms for error estimation and step size adjustment. Higher order simulation methods can also be investigated; however, related research indicates that higher order methods are prohibitively expensive. Development of absolute error bounds for the approximation would also be useful for mission critical systems.

Our exhaustive verification method holds great promise to be able to analyze larger systems as computational power increases. Exciting potential for these methods is held when using multi-core systems and GPUs. Further research could also be completed in variable resolution grids and other computational methods for improving efficiency.

Monte Carlo-based reachability analysis methods using MLS hold a lot of promise for providing reachability and safety results for SHS, and their power could be further enhanced through the development of new boundary placement methods and splitting policies. Breadth first splitting methods and other variance reduction techniques found in related literature could be implemented to be directly compared with our methods as well.

SHS can potentially be used to model biochemical networks with dynamic structures. The dynamic nature of these systems can cause challenges and opportunities for analysis methods to take advantage of the interrelationships within the system.

APPENDIX A

SOFTWARE

Modeling

We developed a modeling framework in software for SHS based on the chemical reaction modeling presented in the modeling chapter. To implement this methodology for simulation we created a scalable model for sequences of chemical reactions and chemical species. Our modeling system requires the input of the chemical system in a text file which specifies the number of chemical species, chemical reactions, and which chemical species are involved in which reactions.

We capture the details of the reactions through a sparse matrix where the columns are the chemical species and the rows are the chemical reactions. If a chemical species a is a reactant in reaction 1, then a -1 is placed in the matrix in position (1,a). If the chemical species is a product then a 1 is placed in the position (1,a). If more than one chemical species is consumed or produced, the integer is placed in the position instead of 1. In this way all possible chemical reactions can be expressed.

Our modeling software uses the input file of the chemical species and reactions combined with information about the reaction rates to generate the simple reaction model in software. This model can me modified to include discrete or probabilistic transitions by including them in the simulation code. Other attributes such as temperature can also be included using simple modifications included in the simulation code.

Simulation

Our simulation software includes C++ implementations of all the simulation methods we presented in previous chapters. The program implements both the EM and MM methods as well as traditional and probabilistic boundary detection and reflection. Adaptive time stepping is also implemented with both error estimation methods allowing the user to choose the appropriate method or combination of methods.

Verification

We have implemented the dynamic programming verification method presented earlier. It allows the user to input a model of up to 8 continuous dimensions, and it is written to use MPI parallelization for up to 32 processors. Because of the complexity of the method our software requires the user to directly manipulate the model in the software. Automated model input methods were tested, but they were found to be too inefficient.

Monte Carlo Methods

Our Monte Carlo software implementation is an extension of our simulation methods. Any simulation method can be used and the approximation statistics are automatically calculated. Our software is designed to be used in both parallel and standalone situations. We use MPI for the parallel implementation.

Summary

We present a comparison and summary of the software developed in Table 16.

Table 16: Software developed

Simulation	-hybrid Euler-Maruyama, first-order -hybrid Milstein method, second-order -adaptive time stepping
Exhaustive Verification	-based on dynamic programming -parallelization
Monte Carlo-based analysis	-reachability or safety -parallelization -multilevel splitting

Computational Resources

We used a standard desktop computer with a 3.0 GHz processor and 1GB RAM for all of the small simulations or scalar verification results. All the parallel results were collected using the computational resources provided by ACCRE at Vanderbilt [5]. Over 2000 processor

[5] http://www.accre.vanderbilt.edu

cores are available through ACCRE of different types and connectivity. We used IBM PowerPC processors with 2.2GHz and 750MB RAM with Myrinet connectivity for low latency.

REFERENCES

[1] A. Abate, S. Amin, M. Prandini, J. Lygeros, and S. Sastry. Computational approaches to reachability analysis of stochastic hybrid systems. In *Hybrid Systems Computation and Control*, volume LNCS. 4416, pages 4–17, 2007.

[2] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. An approximate dynamic programming approach to probabilistic reachability for stochastic hybrid systems. In *IEEE Conference on Decision and Control*, pages 4018–4023, 2008.

[3] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

[4] D. Adalsteinsson, D. McMillen, and T. Elston. Biochemical network stochastic simulator (BioNetS): Software for stochastic modeling of biochemical networks. *BMC. Bioinformatics*, 5(24), 2004.

[5] L. Afzelius, I. Zamora, M. Ridderstrm, T. Andersson, A. Karln, and C. Masimirembwa. Competitive CYP2C9 inhibitors: Enzyme inhibition studies, protein homology modeling, and three-dimensional quantitative structure-activity relationship analysis. *Mol. Pharm.*, 59(4):909–919, 2001.

[6] A. Alfonsi, E. Cances, G. Turinici, B. Di Ventura, and W. Huisinga. Adaptive simulation of hybrid stochastic and deterministic models for biochemical systems. In *ESAIM: Proceedings*, volume 14, pages 1–13, 2005.

[7] R. Alur, C. Belta, F. Ivanicic, V. Kumar, M. Mintz, G. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biomolecular networks. In *Hybrid Systems Computation and Control*, volume LNCS 2034, pages 19–33, 2001.

[8] R. Alur, T. Dang, J. Esposito, Y. Hur, F. Ivančič, V. Kumar, I. Lee, P Mishra, G. Pappas, and O. Sokolsky. Hierarchical modeling and analysis of embedded systems. *Proceedings of the IEEE*, 91(1):11–28, January 2003.

[9] S. Amin, A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Reachability analysis for controlled discrete time stochastic hybrid systems. In *Hybrid Systems Computation and Control*, volume LNCS 3927, pages 49–63, 2006.

[10] M. Apaydin, D. Brutlag, C. Guestrin, D. Hsu, J. Latombe, and C. Varma. Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. *J. of Comp. Bio.*, 10(3-4):257–281, 2003.

[11] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In N. Lynch and B. Krogh, editors, *Hybrid Systems Computation and Control*, volume LNCS 1790, pages 21–31. Springer, 2000.

[12] A. Auger, P Chatelain, and P Koumoutsakos. R-leaping: Accelerating the stochastic simulation algorithm by reaction leaps. *J. Chem. Phys.*, 125:84–103, 2006.

[13] R. Barbuti, S. Cataudella, A. Maggiolo-Schettini, P Milazzo, and A. Triona. A probabilistic model for molecular systems. *Fundamenta Informaticae XX*, pages 1–15, 2005.

[14] C. Belta, P Finin, L. Habets, A. Halasz, M. Imielinski, V. Kumar, and H. Rubin. Understanding the bacterial stingent response using reachability analysis of hybrid systems. In *Hybrid Systems Computation and Control*, volume LNCS 2993, pages 111–125, 2005.

[15] C. Belta, J. Schug, T. Dang, V. Kumar, G. Pappas, H. Rubin, and P Dunlap. Stability and reachability analysis of a hybrid model of luminescence in the marine bacterium vibrio fischeri. *40th IEEE CDC*, Orlando, FL, 2001.

[16] M. Bentele and R. Eils. General stochastic hybrid method for the simulation of chemical reaction processes in cells. *CMSB*, 3082:248–251, 2005.

[17] S. Berman, Halsz, and V. Kumar. MARCO: A reachability algorithm for multi-affine systems with applications to biological systems. In *Hybrid Systems Computation and Control*, volume LNCS 4416, pages 76–89, 2007.

[18] M. Bernadskiy, R. Sharykin, and R. Alur. Structured modeling of concurrent stochastic hybrid systems. *FORMATS LNCS*, 3253:309–324, 2004.

[19] D. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1989.

[20] H. Blom, G. Bakker, and J. Krystul. Probabilistic reachability analysis for large scale stochastic hybrid systems. *IEEE Conference on Decision and Control*, pages 3182–3189, 2007.

[21] H. Blom, J. Lygeros, M. Everdij, S. Loizou, and K. Kyriakopoulos. Stochastic hybrid systems: Theory and safety critical applications. *Lecture Notes in Control and Informations Sciences*, 337, 2006.

[22] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In N. Lynch and B. Krogh, editors, *Hybrid Systems Computation and Control*, volume LNCS 1790, pages 73–88. Springer, 2000.

[23] J. Bucklew. *Introduction to Rare Event Simulation*. New York: Springer-Verlag, 2004.

[24] M. Bujorianu. Extended stochastic hybrid systems and their reachability problem. In R. Alur and G. Pappas, editors, *Hybrid Systems Computation and Control*, volume 2993 of *LNCS*, pages 234–249. Springer, 2004.

[25] M. Bujorianu and J. Lygeros. Reachability questions in piecewise deterministic Markov processes. In *Hybrid Systems Computation and Control*, volume LNCS 2623, pages 126–140, 2003.

[26] M. Bujorianu and J. Lygeros. Theoretical foundations of general stochastic hybrid systems: Modeling and optimal control. In *IEEE Conf. on Dec. and Cont.*, 2004.

[27] Y. Cao, H. Li, and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *J. Chem. Phys.*, 121(9):4059–4067, 2004.

[28] A. Chutinan and B. Krogh. Computational techniques for hybrid system verification. *IEEE J. on Aut. Cont.*, 48(1):64–75, January 2003.

[29] E. Cinquemani, R. Porreca, G. Ferrari-Trecate, and J. Lygeros. Parameter identification for stochastic hybrid models of biological interaction networks. In *Proc. of the IEEE Conf. on Dec. and Cont.*, pages 5180–5185, 2007.

[30] C. Constantini, B. Pacchiarotti, and F. Sartoretto. Numerical approximation for functionals of reflecting diffusion processes. *SIAM J. Appl. Math*, 58:73–102, 1998.

[31] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, July 1995.

[32] D. Darnoko and M. Cheryan. Kinetics of palm oil transesterification in a batch reactor. *JAOCS*, 77(12):1263–1267, 2000.

[33] M. Davis. *Markov Models and Optimization*. Chapman and Hall, 1993.

[34] M. Davis and M. Farid. Piecewise-deterministic processes and viscosity solutions. In W. McEneaney, G. Yin, and Q Zhang, editors, *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W. Fleming*, pages 249–268. Birkhäuser, 1999.

[35] L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In J. Baeten and S. Mauw, editors, *CONCUR'99*, volume 1664 of *LNCS*, pages 66–81. Springer, 1999.

[36] H. de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *J. Comp. Bio.*, 9(1):67–103, 2002.

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

[38] M. Everdij and H. Blom. Piecewise deterministic Markov processes represented by dynamically coloured Petri nets. *Technical Report*, NLR-TP-2000-428, 2000.

[39] M. Farina and M. Prandini. Hybrid models for gene regulatory networks: the case of lac operon in E. coli. In *Hybrid Systems Computation and Control*, volume LNCS 4416, pages 693–697, 2007.

[40] A. Fehnker and F. Ivančič. Benchmarks for hybrid systems verification. In R. Alur and G. Pappas, editors, *Hybrid Systems Computation and Control*, volume LNCS 2993, pages 326–341. Springer, 2004.

[41] S. Fernando, P. Karra, R. Hernandez, and S. Jha. Effect of incompletely converted soybean oil on biodiesel quality. *Energy*, 32:844–851, 2007.

[42] W. Fish and S. Madihally. Modeling the inhibitor activity and relative binding affinities in enzyme-inhibitor-protein systems: Application to developmental regulation in a PG-PGIP system. *Biotechnol. Prog.*, 20(3):721–727, 2004.

[43] W. Fleming and H. Soner. *Controlled Markov Processes and Viscosity Solutions*. Springer-Verlag, 1993.

[44] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *J. Comput. Biol.*, 7:601–620, 2000.

[45] J. Gaines and T. Lyons. Variable step size control in the numerical simulation of stochastic differential equations. *SIAM J. Appl. Math*, 57:1455–1484, 1997.

[46] M. Garvels. *The splitting method in rare event simulation*. PhD thesis, University of Twente, the Netherlands, 1997.

[47] R. Ghosh and A. Arapostathis. Ergotic control of switching diffusions. *SIAM J. on Cont. Optim.*, 35(6):1952–1988, 1997.

[48] R. Ghosh, A. Arapostathis, and S. Marcus. Optimal control of switching diffusions with application to flexible manufacturing systems. *SIAM J. on Cont. Optim.*, 31(5):1183–1204, 1993.

[49] R. Ghosh, A. Tiwari, and C. Tomlin. Automated symbolic reachability analysis with application to delta-notch signaling automata. In *Hybrid Systems Computation and Control*, volume LNCS 2623, pages 233–248, 2003.

[50] R. Ghosh and C. Tomlin. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modeling: Delta-notch protein signalling. *Sys. Bio.*, 1:170–183, 2004.

[51] D. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Phys.*, 22:403–434, 1976.

[52] D. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 115:1716–1733, 2001.

[53] P. Glasserman, P. Heidelberger, and P. Shahabuddin. Multilevel splitting for estimating rare event probabilities. *Operations Research*, 47(4):585–600, 1999.

[54] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. A large deviations perspective on the efficiency of multilevel splitting. *IEEE Trans. on Autom. Cont. AC-43*, 12:1666–1679, 1998.

[55] P. Glynn and D. Iglehart. Importance sampling for stochastic simulations. *Management Science*, 35:1367–1392, 1989.

[56] E. Gobet. Euler schemes and half-space approximation for the simulation of diffusion in a domain. *ESAIM: Probability and Statistics*, 5:261–297, 2001.

[57] M. Griffith, T. Courtney, J. Peccoud, and W. Sanders. Dynamic partitioning for hybrid simulation of the bistable HIV-1 transactivation network. *Bioinformatics: System Biology*, 22(22):2782–2789, 2006.

[58] R. Grosu, S. Mitra, P. Ye, E. Entcheva, I. Ramakrishnan, and S. Smolka. Learning cycle-linear hybrid automata for excitable cells. In *Hybrid Systems Computation and Control*, volume LNCS 4416, pages 245–258, 2007.

[59] A. Halasz, V. Kumar, M. Imielinski, C. Belta, O. Sokolsky, S. Pathak, and H. Rubin. Analysis of lactose metabolism in E. coli using reachability analysis of hybrid systems. *IET Systems Biology*, 1(2):130–148, 2007.

[60] L. Harris and P. Clancy. A partitioned leaping approach for multiscale modeling of chemical reaction dynamics. *J of Chem. Phys.*, 125, 2006.

[61] J. Hespanha. Stochastic hybrid systems: Application to communication networks. In *Hybrid Systems Computation and Control*, volume LNCS 2993, pages 387–401, 2004.

[62] J. Hespanha and A. Singh. Stochastic models for chemically reacting systems using polynomial stochastic hybrid systems. *Int. J. on Robust Cont., Special Issue on Control at Small Scales*, 15:669–689, 2005.

[63] A. Hill, J. Tomshine, V. Wedding, V. Sotiropoulos, and Y. Kaznessis. SynBioSS: the synthetic biology modeling suite. *In press*, 2008.

[64] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. COPASI-a COmplex PAthway SImulator. *Bioinformatics: Systems Biology*, 22(24):3067–3074, 2006.

[65] J. Hu. Application of stochastic hybrid systems in power management of streaming data. In *Proceedings of the American Cont. Conf.*, pages 4754–4759, 2006.

[66] J. Hu, J. Lygeros, and S. Sastry. Towards a theory of stochastic hybrid systems. In *Hybrid Systems Computation and Control*, volume LNCS 1790, pages 160–173, 2000.

[67] J. Hu, M. Prandini, and S. Sastry. Probabilistic safety analysis in three dimensional aircraft flight. In *Proc. of 42nd IEEE Conf. on Decision and Control*, pages 5335–5340, 2003.

[68] J. Hu, W. Wu, and S. Sastry. Modeling subtilin production in bacillus subtilis using stochastic hybrid systems. In *Hybrid Systems Computation and Control*, volume LNCS 2993, pages 417–431, 2004.

[69] HybridSAL. Modeling and abstracting hybrid systems. http://www.csl.sri.com/users/tiwari/HybridSalDoc.ps, 2003.

[70] F. Hynne, S. Dano, and P. Sorensen. Full-scale model of glycolysis in saccharomyces cerevisiae. *Biophysical Chemistry*, 94:121–163, 2001.

[71] K. Jansons and G. Lythe. Exponential timestepping with boundary test for stochastic differential equations. *SAIM J. Sci. Comput.*, 24(5):1809–1822, 2003.

[72] K. Jansons and G. Lythe. Multidimensional exponential timestepping with boundary test. *SAIM J. Sci. Comput.*, 27(3):793–808, 2005.

[73] A. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press, 1970.

[74] A. Julius, A. Girard, and G. Pappas. Approximate bisimulation for a class of stochastic hybrid systems. In *Proceedings of the 2006 American Control Conference*, pages 4724–4729, 2006.

[75] M. Kaern, T. Elston, W. Blake, and J. Collins. Stochasticity in gene expression: From theories to phenotypes. *Nature Reviews*, 6(6):451–464, 2005.

[76] H. Kahn and T. Harris. Estimation of particle transmission by random sampling. *National Bureau of Standards Applied Mathematical Series*, 12:27–30, 1951.

[77] O. Karanfil. A dynamic simulator for the management of disorders of the body water metabolism. Master's thesis, Bogazici University, 2005.

[78] S. Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 1969.

[79] T. Kiehl, R. Mattheyses, and M. Simmons. Hybrid simulation of cellular behavior. *Bioinformatics*, 20(3):316–322, 2004.

[80] P. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, 1999.

[81] X. Koutsoukos. Optimal control of stochastic hybrid systems based on locally consistent Markov decision processes. *Int. J. of Hybrid Sys.*, 4:301–18, 2004.

[82] X. Koutsoukos, P. Antsaklis, J. Stiver, and M. Lemmon. Supervisory control of hybrid systems. *Proceedings of IEEE, Special Issue on Hybrid Systems*, 88(7):1026–1049, 2000.

[83] X. Koutsoukos and D. Riley. Computational methods for reachability analysis of stochastic hybrid systems. In *Hybrid Systems Computation and Control*, volume LNCS 3927, pages 377–391, 2006.

[84] X. Koutsoukos and D. Riley. Safety of stochastic hybrid systems based on discrete approximations. In *IEEE Southeastern Symposium on System Theory*, 2006.

[85] X. Koutsoukos and D. Riley. Computational methods for verification of stochastic hybrid systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 38(2):385–396, 2008.

[86] J. Krystul and H. Blom. Sequential Monte Carlo simulation of rare event probability in stochastic hybrid systems. *16th IFAC World congress*, 2005.

[87] H. Kushner and P. Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer-Verlag, 2001.

[88] M. Kwiatkowska, G. Norman, D. Parker, O. Tymchyshyn, J. Heath, and E. Gaffney. Simulation and verification for computational modelling of signalling pathways. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 1666–1674. Winter Simulation Conference, 2006.

[89] Y. Kwon. Probabilistic modeling and verification of large scale systems. *Technical Report*, UIUCDCS-R-2006-2687, 2006.

[90] H. Lamba. An adaptive timestepping algorithm for stochastic differential equations. *J. of Comput. and Appl. Math.*, 161:417–430, 2003.

[91] G. Lamm. Extended brownian dynamics. III three dimensional diffusion. *J. Chem. Phys.*, 80(6):2845–2855, 1983.

[92] M. Leaning, R. Flood, D. Cramp, and E. Carson. A system of models for fluid-electrolyte dynamics. *IEEE Trans. on Biomed. Eng.*, 32(10):856–64, 1985.

[93] P. L'Ecuyer and B. Tuffin. Splitting for rare-event simulation. *Winter Simulation Conference*, pages 137–148, 2006.

[94] P. Lincoln and A. Tiwari. Symbolic systems biology: Hybrid modeling and analysis of biological networks. In R. Alur and G. Pappas, editors, *Hybrid Systems Computation and Control*, volume LNCS 2993, pages 660–672. Springer, 2004.

[95] J. Lygeros. On reachability and minimum cost optimal control. *Automatica*, 40(6):917–927, 2004.

[96] R. Mannella. Absorbing boundaries and optimal stopping in a stochastic differential equation. *Phys. Lett. A*, 254:257–262, 1999.

[97] X. Mao, G. Yin, and C. Yuan. Stabilization and destabilization of hybrid systems of stochastic differential equations. *Automatica*, 43:264–273, 2007.

[98] I. Marini, L. Bucchioni, P. Borella, A. Del Corso, and U. Mura. Sorbitol dehydrogenase from bovine lens: Purification and properties. *Arch. Biochem. and Biophy.*, 340:383–391, 1997.

[99] M. Mariton. *Jump Linear Systems in Automatic Control*. Marcel Dekker, Inc, New York, 1990.

[100] Mathworks. Simbiology. http://www.mathworks.com/products/simbiology/.

[101] I. Mitchell, A. Bayen, and C. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. on Automatic Control*, 50(7):947–957, 2005.

[102] I. Mitchell and J. Templeton. A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems. In *Hybrid Systems Computation and Control*, volume LNCS 3414, pages 480–94, 2005.

[103] K. Moon, A. Szepessy, R. Tempone, and G. Zourais. Convergence rates for adaptive weak approximation of stochastic differential equations. *Stochastic analysis and applications*, 23(3):511–558, 2005.

[104] E. Mordecki, A. Szepessy, R. Tempone, and G. Zourais. Adaptive weak approximations of diffusions with jumps. *SIAM Journal on Numerical Analysis*, 46(4):1732–1768, 2007.

[105] B. Munsky and M. Khammash. The finite state projection algorithm for the solution of the chemical master equation. *Journal of Chemical Physics*, 124, 2006.

[106] B. Munsky and M. Khammash. A multiple time-step finite state projection algorithm for the solution to the chemical master equation. Technical Report CCDC-06-1130, Center for Control, Dynamical Systems and Computation. University of California at Santa Barbara, 2006.

[107] Y. Noda and S. Sasaki. Regulation of aquaporin-2 trafficking and its binding protein complex. *Biophysica Acta*, 1758:1117–1125, 2006.

[108] H. Noureddini and D. Zhu. Kinetics of transesterification of soybean oil. *JAOCS*, 74(11):1457–1463, 1997.

[109] H. Nurdin. Stochastic hybrid systems: Models and applications. http://citeseer.ist.psu.edu/nurdin01stochastic.html.

[110] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[111] E. Peters and T. Barenbrug. Efficient brownian dynamics simulation of particles near walls. I reflecting and absorbing walls. *Physical Review*, 66:1–7, 2002.

[112] A. Phillips and L. Cardelli. A correct abstract machine for the stochastic pi-calculus. In *LNCS 4230*, 2006.

[113] S. Prajna, A. Jadbabaie, and G. Pappas. Stochastic safety verification using barrier certificates. In *Proc. of 43rd IEEE Conf. on Decision and Control*, pages 929–934, Atlantis, Paradise Island, Bahamas, December 2004.

[114] M. Prandini and O. Watkins. Probabilistic aircraft conflict detection. In *HYBRIDGE, IST-2001-32460*, 2005.

[115] S. Pranja, A. Jadbabaie, and G. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans. on Aut. Cont.*, 52(8):1415–1428, 2007.

[116] C. Priami, A. Regev, E. Shapiro, and W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80:25–31, 2001.

[117] M. Puterman. *Markov Decision Processes-Discrete Stochastic Dynamic Programming*. Wiley, 2005.

[118] M. Ramos, A. Melo, E. Henriques, J. Gomes, N. Reuter, B. Maigret, W. Floriano, and M. Nascimento. Modeling enzyme-inhibitor interactions in serine proteases. *Int. J. Quant. Chem.*, 74(3):299–314, 1999.

[119] D. Riley, X. Koutsoukos, and K. Riley. Reachability analysis of a biodiesel production system using stochastic hybrid systems. In *15th IEEE Mediterranean Conference on Control and Automation*, 2007.

[120] D. Riley, X. Koutsoukos, and K. Riley. Safety analysis of sugar cataract development using stochastic hybrid systems. In *Hybrid Systems Computation and Control*, volume LNCS 4416, pages 758–761, 2007.

[121] D. Riley, X. Koutsoukos, and K. Riley. Verification of biochemical processes using stochastic hybrid systems. *IEEE Multi-conference on Systems and Control*, pages 100–105, 2007.

[122] D. Riley, X. Koutsoukos, and K. Riley. Modeling and simulation of biochemical processes using stochastic hybrid systems: The sugar cataract development process. In *Hybrid Systems Computation and Control*, volume LNCS 4981, pages 429–442, 2008.

[123] D. Riley, X. Koutsoukos, and K. Riley. Simulation of stochastic hybrid systems with switching and reflecting boundaries. *Winter Simulation Conference*, Miami, FL, 2008.

[124] D. Riley, X. Koutsoukos, and K. Riley. Modeling and analysis of the sugar cataract development process using stochastic hybrid systems. *IET: Systems Biology*, Accepted, 2009.

[125] D. Riley, X. Koutsoukos, and K. Riley. Reachability analysis for stochastic hybrid systems using multilevel splitting. In *Hybrid Systems Computation and Control*, 2009.

[126] D. Riley, X. Koutsoukos, and K. Riley. Reachability analysis of stochastic hybrid systems: A biodiesel production system. *Euopean Journal of Control: Special Issue on Stochastic Hybrid Systems*, page Submitted, 2009.

[127] R. Rubinstein and D. Kroese. *Simulation and the Monte Carlo Method*. Wiley, 2007.

[128] H. Salis and Y. Kaznessis. Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *J. Chem. Phys.*, 122:54–103, 2005.

[129] H. Salis, V. Sotiropoloulos, and Y. Kaznessis. Multiscale Hy3S: Hybrid stochastic simulation for supercomputers. *BMC Bioinformatics*, 7(93), 2006.

[130] I. Shmulevich, E. Dougherty, S. Kim, and W. Zhang. Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, 2001.

[131] S. Strubbe, A. Julius, and A. van der Schaft. Communicating piecewise deterministic Markov processes. In *Proc. IFAC Conf. Analysis and Design of Hybrid Systems - ADHS03*, pages 349–354, June 2003.

[132] S. Strubbe and A. van der Schaft. Stochastic semantics for communicating piecewise deterministic Markov processes. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 6103–6108, 2005.

[133] A. Szepessy, R. Tempone, and G. Zouraris. Adaptive weak approximation of stochastic differential equations. *Comm. Pure Appl. Math*, 54:1169–1214, 2001.

[134] R. Thomas. Regulatory networks seen as asynchronous automata: A logical description. *J. Theor. Biol.*, 153:1–23, 1991.

[135] C. Tomlin, I. Mitchell, A. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, July 2003.

[136] L. Tournier and E. Farcot. Hybrid model of gene regulatory networks, the case of lac operon. *Rapport LMC-IMAG*, 2002.

[137] M. Troyer, B. Ammon, and E. Heeb. Parallel object oriented Monte Carlo simulations. *ISCOPE*, LNCS 1505:191–198, 1998.

[138] P. Ye, E. Entcheva, S. Smolka, M. True, and R. Grosu. Hybrid automata as a unifying framework for modeling cardiac cells. In *In Proc. of EMBS*, pages 4151–4154, 2006.

[139] Y. Zhang, M. Dube, D. McLean, and M. Kates. Biodiesel production from waste cooking oil: 1 process design and technological assessment. *Bioresource Technology*, 89:1–16, 2003.