1-1-2011

Adaptive Methods For Stochastic Simulation Of Biochemical Systems

Alexandra Teslya
Ryerson University

Adaptive methods for stochastic simulation of biochemical systems

by

Alexandra Teslya

Bachelor of Science, York University, 2009

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Applied Mathematics

Toronto, Ontario, Canada, 2011

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Adaptive methods for stochastic simulation of biochemical systems

Master of Science 2011

Alexandra Teslya

Applied Mathematics

Ryerson University

Computational and Systems Biology are recently experiencing a rapid development. Mathematical modeling is a key tool in analyzing critical biological processes in cells and living organisms. In particular, stochastic models are essential to accurately describe the biochemical processes in the cell. However, stochastic models are computationally much more challenging than the traditional deterministic models. Also, the sheer scale of biological processes makes efficiency of their simulation a key issue. In this thesis we study the numerical solution of a continuous stochastic model of biochemical systems, the Chemical Langevin Equation. We propose an adaptive step-size method for the Euler-Maruyama scheme applied to small noise problems. The adaptive technique is p-mean convergent and computes simultaneously all the trajectories, using the same time-step on all trajectories. Our adaptive algorithm is tested on several examples arising in applications and shown to perform much better than the fixed step-size schemes.

Acknowledgements

I would like to thank everyone who made this thesis possible. First and foremost, my supervisor Dr. Silvana Ilie who is a great scientist, teacher and mentor. I could not have written this thesis without her encouragement and guidance. I would also like to thank my family and Nick Yakovets for their support in many ways during these times. Finally, I owe the greatest debt of gratitude to the examining committee members, Dr. Jean-Paul Pascal, Dr. Katrin Rohlf and Dr. Marcos Escobar for sparing their time to read this thesis and to be present at its defense.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years there has been a great interest in mathematical modeling and simulation in the rapid rise of Computational and Systems Biology [13],[20]. Mathematical modeling in biology can be employed in various scales of precision (ranging from molecule interactions to population dynamics) for a number of different purposes (ranging from simulation of the behavior to assessing the risks associated with particular states of a system). It allows for the in silico prediction of the organism or the population dynamics in the future, thus foregoing computational and time costly experiments.

Naturally, a biological system can not be depicted in all its complexity by mathematical models, since such a model will be hard to analyze. One needs to have a number of simplifying assumptions that are close to reality in order to reduce the dynamics of the system to a relatively small number of interacting modules. The abstraction allows to translate a biological process into a mathematical model and to analyze the resulting model using a highly developed mathematical theory. Mathematical and computational results of the analysis can be interpreted into biological terms. In this way we can obtain, for example, insight on some aspects of gene interactions, without having to model the whole process. Breaking a complicated system into a number of modules and analyzing these modules individually will provide for a better and more systematic understanding of the whole organism dynamics. Having understood the dynamics of simpler models, we can refine and extend them to make the modeling more realistic and accommodating to more sophisticated inquiries. Biological processes that can benefit from mathematical modeling range from population dynamics to mitosis (division) of an individual cell. Each level is subject to the previously collected biological information and patterns, as well as to some degree

of abstraction. Having studied a sub-component, we are able to discern the behavior of the network that enlists it, by facilitating our knowledge of individual component reactions to a particular event. Mathematically, this is achieved via reduction of complicated processes to a parametrized model, where the major components whose interactions are being analyzed are depicted as interacting compartments (rather than complicated systems in their own right). Another reason for abandoning the modeling of a system in the full detail is the lack of all the necessary quantitative data, such as the rates of the processes. On the other hand, if there is a collected data about the average behavior of a component, we can come up with a more detailed and complicated picture of its dynamics.

For example, a century ago an eukaryotic cell was viewed as a sack filled with liquid protoplasm. Currently it is known that a complicated molecular structure, called cytoskeleton, heavily influences the cell dynamics [12]. Knowing how the cell structure looks in general, we can deduce which integral chemical processes influence its dynamics, shape and internal chemistry due to the structure.

In this thesis we will focus on the stochastic numerical simulation of biochemical processes. This approach, guided by pre-existing knowledge of the structure of the problem, allows for the quantitative analysis of some key biological networks, and in particular of the gene regulatory networks. Below we discuss the advantages and the challenges of stochastic modeling using examples in genetic networks and some critical biological processes.

To date, due to the development of experimental techniques and of genetics and molecular biology, many studies have been conducted to detect and gather information about genes and their regulatory sites for various species of living organisms. It is extremely important to gain the understanding based on the above knowledge about the gene interaction. Thus, tools for organization and analysis of the collected data are in high demand. While certain patterns in molecular dynamics through which regulation is executed have been detected, the larger dynamics of regulatory networks still remains a great challenge.

One of the areas that have already benefited from mathematical modeling and that anticipates even greater development is genetics. In general, the genome of an organism has many important functions and plays a central role in many cellular processes like the cell responses to environmental stimulation, the replication of the DNA prior to cell division, and many more. Therefore, gene expression is a complex process that has to be regulated at many stages. This complexity prompts the existence of genetic regulatory systems that are structured in accordance with the complex interactions of DNA, RNA, proteins and other molecules [13]. To benefit from the gain of experimental methodology, theoretical modeling of

the biological processes has been employed. The models are usually built using some understanding of the gene interactions and the regulation of the system dynamics under certain conditions. These models would allow (ideally) the use of the collected data to fill out the gaps in the knowledge of the regulatory system dynamics (occasionally foregoing costly experiments), as well as to understand its structure and to make forecasts of its behavior. The predictions of the model are verified against the available data for possible refinement of the system parameters. The development of tools that will allow for scientists without a significant mathematical background to build and analyze the regulatory systems' models behavior can bring drastic advancement in the field of genetics.

An example of a process relatively common to many species, that could benefit from the insight of stochastic modeling is the *eukaryotic cell division cycle*. It is a very complex process which involves many aspects, such as the cell having to ensure that prior to division both of its copies (daughter cells) have all the necessary components and information to function on their own. This means the cell has to reproduce its components. This process is guided by a complex network of interacting genes and proteins (called cyclin-dependent kinases). While this process is deterministic in many aspects, there are occasional fluctuations from the average behavior pattern. Fluctuations may arise in the total time it takes a cell to prepare for division and to divide, on the daughter cell sizes after the division, as well as in the distribution (occasionally uneven) of the mRNAs and proteins to the daughter cells. The size of the cell is important since it is in direct correlation with the time it takes the cell to divide. That is, the smaller the cell at birth, the larger is the time of the cell division cycle. Clearly, these aspects influence the size of the proliferating cell populations. While certain patterns of what is happening in the general case are established, there exist deviations from the average, which may influence the whole system dynamics. These fluctuations may be explained by the inherent randomness of the system that controls the division of a cell. Another source of probabilistic factors is the small number of very important particles (e.g, the mRNAs in cycle progressions of the yeast population). The fluctuations in the dynamics of these species will be crucial for the process [14]. The cell division cycle is an excellent example of a system that could benefit from incorporating the random effects into its model. On the other hand, this is an example of the lack of necessity to trace all the aspects of the cell functioning in order to simulate its dynamics of the division cycle. Rather, we can concentrate on such pivotal issues as the molecular interactions (mRNA and any number of proteins, depending on the grain of the model), the cell size and the cell cycle. This effectively reduces the system with respect to the modeling

parameters.

Another example where stochastic modeling is useful are the *circadian rhythms*. As a living organism needs to keep up with the periodic nature of the environment, many organisms have developed a way to maintain the nature's rhythms. The mechanism that is responsible with this is called a circadian clock. Usually, it has a period of roughly 24 hours. During this cycle, the biochemical processes in the bodies are adjusted to the needs of the organism according to the time of the day. These rhythms have a periodic nature with fixed amplitude and maintain this property. On the other hand, the circadian rhythms are subject to some biochemical processes, which are extrinsically vulnerable to the outside environment factors (such as the temperature) and intrinsically are inherently stochastic. As in any systems that are governed by chemical reactions, this system is easily disturbed if there are fluctuations in some important species of molecules that have small population numbers. However, the nature found a way to control these occurrences. To study the periodic nature of circadian rhythms, stochastic systems should be used, since they will allow to incorporate the random effects described above. Thus, stochastic modeling gives a more realistic picture of such processes [21].

Another example where stochastic modeling proved to be relevant is the analysis of the *Phage* λ-*Infected Escherichia coli cells* [1]. The stochastic approach was suitable as there existed unexplained fluctuations in the rates of gene expression that were observed experimentally. These fluctuations caused unevenness in the time of protein production. This phenomenon affected the individual cells, as well as it had a profound effect on the total cell populations dynamics. If these proteins were in low concentrations in the system and competitively control a switch path, then a variation in their quantity could turn initially homogeneous system of cells into two completely different subpopulations.

In the simulation of biological processes that have every molecular population very large, the traditional deterministic system of reaction rate equation, which operates with molecular concentrations, is a good enough representation. However, in cellular biology there are processes (such as the regulation of gene expression, where the gene transcription factors interact with the DNA binding sites) that have their most significant molecules in small population numbers. It has been observed that in the system where certain populations are hardly larger than tens, it is more suitable to operate with discrete numbers of molecules, rather than with their concentrations. Hence, the need for considering more general, discrete models arises [20]. On the other hand, in cellular processes random fluctuations due to the presence of small populations of molecules can no longer be neglected. This is particularly true when

dealing with species which are key to the dynamics of the whole system [20].

The stochastic modeling of chemically reacting systems was pioneered by Daniel Gillespie [8], [7]. Gillespie introduced the Stochastic Simulation Algorithm (SSA) to simulate the solution of the Chemical Master Equation (CME), which is the most refined mathematical model of well-stirred bio-chemical kinetics. The algorithm is a Monte Carlo type method, which provides numerical realizations of the state of the system and whose statistics are consistent with the CME. The algorithm simulates one reaction at a time, by generating the time and the index of the next reaction conditional on the current state of the system. While this algorithm is exact, it is extremely slow on most realistic applications. A number of extensions and approximations were developed based on various properties of the biochemical systems under consideration [19], [10], [3].

One of the core problems for SSA is that often biochemical systems have high probabilities of some reactions taking place. Then, it is very time and resource consuming to use Gillespie's algorithm even during a relatively small time interval. Thus, Gillespie proposed in [9] that instead of simulating one reaction at a time, to update the state of the system after some predefined time leaps. More precisely, the interval of integration is divided into sub-intervals. We simulate the number of reactions that fired during each time interval by a Poisson distributed variable and update the state variable at the end of each interval correspondingly. This algorithm assumes that the probability of each reaction taking place does not change significantly over the time leap.

For systems where all the reactions have high probabilities of taking place there is a further simplification. The Poisson distributed variables with large means can be well approximated by normally distributed variables with appropriate means and variances. This allows for the reduction of the Chemical Master Equation model to the simplified Chemical Langevin Equation (CLE). In addition, it leads to the Langevin Leaping algorithm (a direct result of solving the CLE using the Euler-Maruyama method for stochastic differential equations). Similarly to the tau-leaping algorithm before, this algorithm requires that the probability of each reaction taking place in a time interval does not change significantly over that time leap. It is faster and usually admits larger time steps, due to the condition that many reactions of each type occur during a time step.

However, systems arising in applications are rarely 'well-behaved'. Often their models are stiff. Stiff systems are characterized by varying scales of the reaction rates, the fastest of which is stable. The slow reactions often happen to be the key reactions in the system. Among the algorithms designed to deal

with stiffness are the slow scale Stochastic Simulation Algorithm [3] and the implicit tau-leaping method [16].

Another issue with the approximate algorithms for the CME, such as the tau-leaping algorithms, are the negative populations occasionally arising due to the unrestricted nature of the Poisson random variables. This aspect was addressed directly by replacing the Poisson distributions by the Binomial distributions [19]. Another approach is to follow the so called critical reactions (reactions that are few firings away from depleting the population of some species of molecules) [2] and to impose more strict Leap conditions.

Over the years there have been proposed improved ways to impose for the Leap Condition. The change in the individual reactions propensities over one step were restricted in terms of the largest reaction propensity, or of the sum of all propensities. Another method proposed was that the relative change in each reaction propensity was negligible over one time step. Adaptive step size methods proved to be useful in dealing with this challenge. Intuitively, a time step of the algorithm should be adapted to the significant changes in the propensity functions.

As was pointed out by Sotiropoulos and Kaznessis [18] stiff systems are prime candidates for adaptive time step algorithms. Since, if we restrict the step size to a single value, it will have to be very small on all regions of integration to ensure that the accuracy requirement is met, although some regions would allow for larger time steps. In this thesis, we study the numerical solution of the Chemical Langevin Equation based on the Euler-Maruyama method. We develop an adaptive time-stepping strategy for this method. One of the issues that arises with the adaptive time stepping is how to deal with step rejections, when the accuracy criteria is not satisfied. During integration, we generate Brownian paths, which are based on time steps. Once we reject a time step, if we reject the associated Brownian path point, then we will skew the distribution of the numerical solution. Instead, we chose to generate intermediate Wiener increments for the smaller time steps, that are conditioned on the previously generated points. This will ensure that we maintain the correct distribution.

In this thesis we offer an approach based on the p-mean of the local discretization error, that makes the correction of the leaping step after the perceived error across all trajectories has been evaluated. This approach has the advantage that it does not differentiate between the states of the system (equilibrium or transient), which would require for the step to grow larger or smaller. It treats all reactions uniformly. This time stepping strategy was proposed for stochastic differential equation models by Romisch and

Winkler [17]. Their method employed the leading term of the estimated p mean local error for controlling the step size. The authors proved that the Euler-Maruyama scheme for the numerical solutions of SDEs is p mean stable. This result directly applies to our method. The nature of the p-mean analysis gives a new strategy of trajectory generation, which results in identical stepping sequences for all trajectories. Such an approach allows to estimate the error on the collection of data from all trajectories. This leads to a speed up in computations.

This algorithm guarantees that the accuracy requirement is satisfied on each time step. We expect significant improvement in terms of the computational speed, since this method estimates the local error based on the information provided by the collection of trajectories.

Finally, we provide a short overview of this thesis. In Chapter 2, we introduce the mathematical models and simulation methods of biochemical systems. In this chapter, we present the stochastic simulation algorithm and the condition under which it is possible to develop much faster tau-leaping algorithms. We discuss the underlying mathematical theory for stochastic differential equations (SDE). We describe a particular numerical method to solve the SDEs, the Euler-Maruyama scheme, that we use in this thesis. In Chapter 3, we introduce an adaptive stepping strategy for the numerical approximation for the solution of an SDE, and we show how it applies to the Chemical Langevin Equation. We discuss the theoretical foundation of the method, as well as show that the Euler scheme is p-mean stable. In Chapter 4, we test on three biochemical reacting systems both the adaptive and the fixed step size algorithms and showcase the advantages of the proposed adaptive strategy. In Chapter 5, we present our conclusions and discuss further improvements and extensions of the method.

Chapter 2

Background

Many important biological processes and in particular cellular processes can be described as systems of biochemical reactions. The most general mathematical model of such a system is given by the Molecular Dynamics. The Molecular Dynamics model keeps track of the location and velocity for each molecule in the system, at any given time. However, for most practical applications in Systems Biology it is computationally very expensive to approximate numerically the solution of the Molecular Dynamics model. A great simplification of this model is obtained under the mild assumption that the system is well-stirred, i.e. spatially homogeneous. We also assume that the biochemically reacting system is held at a constant temperature, in a constant volume Ω. Then the system can be described by tracing the time evolution of the number of molecules of each reacting species.

Below, we will present the mathematical framework that gives the most refined model of the dynamics of biochemical systems satisfying the above assumptions.

The set S of interacting molecules in the system can be partitioned into N subsets: $[S_1, S_2, ..., S_N]$, where $N \geq 1$ is the number of different molecular species in the system. Thus, $S = \bigcup_{i=1}^{N} S_i$ and $\forall i, j, S_i \bigcap S_j = \emptyset, i \neq j$. These molecules interact according to a finite set of reactions $[R_1, ..., R_M]$, where $M \geq 1$ is the number of biochemical reactions occurring in the system.

Each reaction is assumed to be an instantaneous event, that is it takes the smallest measurable unit of time. At each moment in time, the system can be described by the vector $X(t) = (x_1(t), x_2(t), ...x_N(t))^T$, where we denoted by $x_i(t)$ the number of molecules of species S_i present in the system at time t. Thus, X(t) is a snapshot of the system in terms of the molecular population sizes in time.

We would like to trace the dynamics of X(t) in time. We note that X(t) is a stochastic process. As we shall see below, stochastic mathematical models are essential to accurately describe the evolution of biochemically reacting systems, in particular when some reactant species have low population numbers. For example, at the level of a single cell, the DNA and RNA have only few copies. In such cases the traditional deterministic models of reaction rate equations fail to correctly predict the dynamics of the system. Unlike the deterministic models, the stochastic model operates with quantities of molecules, rather than their concentrations in the given volume.

The chemical reactions are usually represented as:

$$S_1 + S_2 \xrightarrow{c} S_3 + S_4 \ . \tag{2.1}$$

The molecules on the left hand side of the equation are called reactants, and the molecules on the right hand side are called products of the reaction. Each reaction R_j has a corresponding reaction rate c_j. Note, that a rate for unimolecular reaction has a different meaning than a rate of a bimolecular reaction.

In what follows we shall consider all the chemical reaction rate constants scaled in terms of the numbers of molecules, rather than of the concentrations.

Every reaction channel R_j can be described explicitly by two quantities: the reaction propensity function $a_j(X(t))$, where X(t) is the current state of the system and the state-change vector v_j:

- For every reaction R_j, the propensity function $a_j(x)$ is defined such that $a_j(x)\mathrm{d}t$ gives the probability that a single reaction R_j will take place in the infinitesimal time interval $[t, t + \mathrm{d}t)$, provided that at time t, $X(t) = x$.

- Every reaction has reactants and product molecules. As the reaction channel fires, the populations of both the reactants and the products are altered. The state change vector v_j describes the change in the population of all types of molecules in the system after one reaction R_j occurs. Thus, if at time t the state of the system was $X(t) = x$ and one reaction R_j fired over the time interval $[t, t + \mathrm{d}t]$, then the state of the system at time $t + \mathrm{d}t$ is $X(t + \mathrm{d}t) = x + v_j$. We denote by $v_j = [v_{1j}, v_{2j}, ..., v_{Nj}]^T$, where v_{ij} is the change in the number of species S_i if one reaction R_j occurred. The matrix $V = (v_{ij})_{1\leq i\leq N, 1\leq j\leq M}$ is commonly referred to as the *stoichiometric matrix*.

Propensity functions In this thesis we only consider systems with either unimolecular and bimolecular reactions. This is due to the reaction 'rapidity' assumption made earlier. Indeed, provided that a reaction takes place in an infinitesimal time interval, then no more than two molecules can interact.

It is obvious from its definition that a propensity function depends on both quantities of reactants present in the system and on the reaction rate of the equation.

- *Unimolecular reactions:*

$$S_1 \xrightarrow{c_j} products \tag{2.2}$$

Recall, that the definition of the propensity function states that $a_j(x)dt$ is the probability that the reaction R_j will take place in the time interval $[t, t + \mathrm{d}t]$ provided that at time t, the state of the system was $X(t) = x$. On the other hand, the probability that a unimolecular reaction will take place is directly proportional to the quantity of the appropriate reactant molecules in the system. Hence the propensity of (2.2) is:

$$a_j(x)dt = c_j x_1(t)dt$$

- *Bimolecular reactions:*

Bimolecular reactions can be of two types: the reactant molecules are of the same species or the reactant molecules belong to different species. The probability that such a reaction will take place is proportional to all possible distinct combinations of molecules of reactants, with the reaction rate being the constant of proportionality. Let us consider a reaction with reactants of the same species:

$$S_1 + S_1 \xrightarrow{c_j} products \tag{2.3}$$

If the reactant molecules are of the same type, then the total number of combinations is $(x_1-1)x_1/2$, thus the propensity of (2.3) is

$$a_j(x) = \frac{c_j}{2}(x_1 - 1)x_1 \tag{2.4}$$

Let us assume that the reaction has different reactant species:

$$S_1 + S_2 \xrightarrow{c_j} products \tag{2.5}$$

Since the molecules are of different types, the total number of combinations is $x_1 x_2$ and thus the reaction (2.5) has propensity:

$$a_j(x) = c_j \cdot x_1 \cdot x_2 \ .$$

According to their definition, the propensity functions are probability based functions, and therefore the most suitable way to describe a chemically reacting system is to trace the evolution of the conditional probability function $P(x, t|x_0, t_0)$ in time. We denoted by $P(x, t|x_0, t_0)$ the probability of the system to be in state x at time t, $X(t) = x$, given that the initial state of the system was x_0, $X(t_0) = x_0$. Moreover, we observe that $X(t)$ is a Markov process. Indeed, the state of the system at time $t + \mathrm{d}t$ depends only on the state of the system at time t.

2.1 Chemical Master Equation

The state of the system at the initial time t_0 is given by $X(t_0) = x_0$. We take $\mathrm{d}t$ to be small enough that only one reaction can take place during the time interval $[t, t + dt]$. There are two possible situations which lead to the system being in the state x at the time $t + dt$ provided that $X(t_0) = x_0$. These two possibilities are mutually exclusive and cover the space of all possibilities exhaustively. Either the system is already at state x at time t, and then the probability that it is in state x at time $t + dt$ is equal to the probability of no reaction taking place during the interval $[t, t + dt)$. The corresponding probability is:

$$P(t, x|x_0, t_0)(1 - \sum_{j=1}^{M} a_j(x)dt) \tag{2.6}$$

Or, the system is one reaction away from the state x at time t, and that reaction occurs in $[t, t + dt)$. Thus, summing over the probabilities for all reactions which may take place in the system we obtain:

$$\sum_{j=1}^{M} P(x - v_j, t - \Delta t|x_0, t_0)a_j(x - v_j)dt \tag{2.7}$$

Therefore, by exhausting the space of all possibilities we derive:

$$P(x, t + dt|x_0, t_0) = P(x, t|x_0, t_0)(1 - \sum_{j=1}^{M} a_j(x)dt) + \sum_{j=1}^{M} P(x - v_j, t|x_0, t_0)a_j(x - v_j)dt \tag{2.8}$$

By extracting $P(x,t|x_0,t_0)$ from both sides of the equation and then dividing both sides by dt, we obtain the following result:

$$\frac{P(x,t+dt|x_0,t_0)-P(x,t|x_0,t_0)}{dt}=\sum_{j=1}^{M}\left[P(x-v_j,t|x_0,t_0)a_j(x-v_j)-P(x,t|x_0,t_0)a_j(x)\right]. \quad (2.9)$$

Taking the above equation to the limit as $dt \to 0$, we obtain the following equation:

$$\frac{dP(x,t|x_0,t_0)}{\mathrm{d}t}=\sum_{j=1}^{M}P(x-v_j,t|x_0,t_0)a_j(x-v_j)-P(x,t|x_0,t_0)a_j(x) \quad (2.10)$$

The equation (2.10) is called the Chemical Master Equation (CME) and it is a system of ordinary differential equations (ODEs), with one equation for each possible state of the system (i.e every possible combination of molecules in the reaction system under consideration). Its solution gives the evolution of the probability of the system being in a particular state, given its initial state x_0 at time t_0. Note, that for a system with many molecules and many reactions, the size of this system is very large. This happens in most applications. Except for cases of systems which are relatively simple (i.e. for which both the number of species and of reactions is small), it is impossible to solve CME analytically.

2.2 Stochastic simulation algorithm

There is a Monte Carlo type algorithm developed by Daniel Gillespie [8], which allows the computation of the realizations of trajectories in exact accordance with the statistics as given by the CME. Thus, having generated a sufficiently large number of trajectories, we can approximate the mean, the variance and other moments for the solution of the CME model. The algorithm is called the Stochastic Simulation Algorithm (SSA).

The key concept in deriving the algorithm is the conditional probability function, $p(\tau,j|x,t)$. We define $p(j,\tau|x,t)d$t to denote the joint probability that, provided that the system's state at time t is $X(t)=x$, the next reaction will take place in the infinitesimal interval $[t+\tau,t+\tau+\mathrm{d}t)$ and the index of this reaction will be j. By definition, this probability is the joint probability density function for the next reaction index and the time to the next reaction.

We use the following notation:

- $p(j, \tau|x, t)\mathrm{d}\tau$ is the probability that, provided that the system is in state x at time t, the next reaction will occur in the time interval $[t+\tau; t+\tau+\mathrm{d}\tau)$ and will be the reaction j. Note, that here $\mathrm{d}\tau$ is infinitesimal, so that only one reaction can take place during $\mathrm{d}\tau$.

- $P_0(\tau|x, t)$ is the probability that, if system is at the state x at the time t, no reaction will fire during the time interval $[t, t+\tau)$.

In order for the reaction j to be the next reaction that will change the state of the system and for it to fire in the time interval $[t+\tau; t+\tau+d\tau)$ the following two events must happen: (1) No reaction fires during the time interval $[t; t+\tau)$ and (2) The reaction j fires during the time interval $[t+\tau; t+\tau+\mathrm{d}\tau)$. These two probabilities are denoted by $P_0(\tau|x, t)$ and $a_j(x)d\tau$, respectively. Hence, the expression for the conditional probability $p(j, \tau|x, t)d\tau$ is:

$$p(j, \tau|x, t)d\tau = P_0(\tau|x, t)a_j(x)d\tau \ . \tag{2.11}$$

On the other hand, the event that no reaction takes place during the time interval $[t; t+\tau+\mathrm{d}\tau)$ is the combination of the two events: (1) No reaction takes place during the time interval $[t; t+\tau)$ and (2) No reaction takes place in the time interval $[t+\tau; t+\tau+\mathrm{d}\tau)$. Therefore we obtain:

$$P_0(\tau+\mathrm{d}\tau|x, t) = P_0(\tau|x, t)(1 - \sum_{i=1}^{M} a_i(x)\mathrm{d}\tau) \ . \tag{2.12}$$

Subtracting $P_0(\tau|x, t)$ from the both sides of the equation 2.12, dividing both sides by $\mathrm{d}\tau$ and taking the limit as $\mathrm{d}\tau \to 0$, leads to the following differential equation for $P_0(\tau|x, t)$:

$$\frac{\mathrm{d}P_0(\tau|x, t)}{\mathrm{d}\tau} = -P_0(\tau|x, t)\sum_{i=1}^{M} a_i(x) \ . \tag{2.13}$$

Introducing the following notation $a_0(x) = \sum_{i=1}^{M} a_i(x)$, we see that the above equation is an ordinary differential equation (ODE). Its solution is:

$$P_0(\tau|x, t) = \exp(-a_0(x)\tau) \ . \tag{2.14}$$

Substituting the expression for $P_0(\tau|x, t)$ in (2.14) into the equation (2.11) gives us the following

joint probability density function for the index j of the reaction as well as for its time of occurrence, τ:

$$p(j, \tau|x, t)\mathrm{d}\tau = \exp(-a_0(x)\tau)a_j(x) \ . \tag{2.15}$$

The equation (2.15) can be written as:

$$p(j, \tau|x, t) = [a_0(x)\exp(-a_0(x)\tau)]\,\frac{a_j(x)}{a_{0(x)}} \ . \tag{2.16}$$

Thus, the joint probability density function is a product of the probability density function for the time of occurrence and of the probability density function for index of the next reaction. To calculate the time to the next reaction, we draw a sample of an exponentially distributed random variable with mean equal to $1/a_0(x)$.

To calculate the index of the next reaction, we draw a sample of a discrete random variable, where the chance of the j-th reaction taking place is proportional to the value of $\frac{a_j(x)}{a_0(x)}$ for $1 \leq j \leq M$. This leads to the following Stochastic Simulation Algorithm:

1. Set i=1, set the initial state $X(0) = x_0$ and the initial time t = t_0
2. Draw two samples of the uniformly distributed unit random variable, r_1 and r_2.
3. Evaluate all the propensity functions at time t.
4. Calculate the time to the next reaction, τ, by using the following equation

$$\tau = \frac{1}{a_0(x)} ln(\frac{1}{r_1})$$

5. Calculate the index of the new reaction to be smallest j such that the cumulative sum $\sum_{i=1}^{j} a_i(x) > r_2 a_0(x)$.
6. Update the time variable t to be $t = t + \tau$
7. Update the state of the system at time t, $X = X + v_j$
8. Repeat steps 2-5 until $t > T_f$. Here T_f is some pre-defined stopping time.

This algorithm provides accurate realizations of the trajectories described by the CME, that is the resulting statistics of these realizations coincide with the CME solution statistics. It has been successfully employed for many models of biochemical systems, in particular for the models of λ-phage [1] and that of circadian rhythms [6].

If the trajectories obtained by using the SSA are similar to those obtained by numerically solving the corresponding reaction rate equations (RREs), we can conclude, that the fluctuations in the system are negligible. However, if the trajectories obtained by the SSA are different from those obtained from solving the corresponding RRE, we conclude that the RRE model does not provide an accurate description of the system dynamics and solving it may lead to an inaccurate representation of the system. However, in the systems where reactions rates are relatively large, it is very expensive to use the SSA, since the reactions rates contribute to the propensity function sizes, and consequently to their sum. We remember that the inverse of this sum is the expected value for the intervals between consecutive reactions. Consequently, the sum is large, the time interval between two consecutive reactions is very small. Therefore, when the propensity functions are large, the number of reactions in the system is large. Thus, it is time consuming to produce a simulation of a system involving a very large number of reactions occurring in short intervals of time, by simulating each reaction in the systems as the SSA does. Another issue with using the SSA is the necessity to update the propensity functions after each reaction occurs, which is very time consuming in systems with many reactions. Alternative methods are required. Such methods sacrifice a certain degree of accuracy, in return for a gain in the efficiency of the algorithm. These algorithms are valid under certain conditions. We will discuss these conditions in the following sections.

2.3 Tau-leaping method

The key idea of this algorithm is to not simulate each reaction in the system, but rather to record the system's state after a predicted time step. Thus, the system state progresses in time by time leaps, while recording the number of times each reaction fired after each leap. This approach is valid under the following condition:

Leap condition We assume that the size of a time step is sufficiently small, so that no propensity function changes in a 'significant' manner during that time step.

There are several ways to ensure the invariability of the propensity functions. We will return to this criterion later. We mention that, it is assumed that the time step is small enough to treat the propensity functions as constants over each leap.

The Leap condition allows us to treat $a_j(x)\tau$ as a constant over the time leap τ. Thus, $a_j(x)\tau$ is constant during the interval $[t; t+\tau]$, $\forall j = 1, .., M$ and therefore $a_j(x)\tau$, the probability that a reaction j will fire during the time interval remains constant. The Poisson random $P(\lambda, \tau)$ variable is defined as the number of times the event will occur in the time interval of length τ, provided that the probability that an event will occur in the infinitesimal interval $d\tau$ is $\lambda d\tau$. (Here λ is a non-negative constant). Therefore, the number of times each reaction j fires during the time leap can be approximated by drawing a sample of a Poisson distributed random variable with mean and variance equal to $a_j(x)\tau$. Knowing the state of the system at the time t, we can approximate the state of the system at time $t + \tau$ by simulating the number of times each reaction fired in this interval and updating the state system correspondingly. Note that for now, we assume that there exists a procedure for finding the step size τ, which satisfies the Leap Condition and that we are using such a τ. We will describe this in more detail in the later chapters.

Let us define K_j as the number of times the reaction j fired during $[t; t + \tau)$. Then:

$$X(t + \tau) = X(t) + \sum_{j=1}^{M} v_j K_j \ . \tag{2.17}$$

Here v_j is j-th column of stoichiometric matrix V, depicting the change in the system state due to the occurrence of the reaction j. Since $a_j(x)$ is assumed to be constant, we approximate K_j on $[t, t + \tau)$ by:

$$K_j = P(a_j(x)\tau) \tag{2.18}$$

Note that $P(a_j(x)\tau)$ are independently distributed Poisson random variables. And thus substituting (2.18) into (2.17), we obtain the following equation:

$$X(t + \tau) = X(t) + \sum_{j=1}^{M} v_j P(a_j(x)\tau) \ , \tag{2.19}$$

This equation is key to the tau-leaping algorithm used to approximate the solution of a system whose

dynamics is described by the CME. The tau-leaping algorithm is as follows:

1. Set $t = t_0$ and $X(t) = X_0$
2. Calculate all the propensity functions.
3. Pick τ compatible with the Leaping condition.
4. For $1 \leq j \leq M$, calculate the number of times each reaction has fired during the time interval $[t, t + \tau)$, using
$$K_j = P(a_j(X(t))\tau) \ .$$
5. Update the system state variable $X(t + \tau) = X(t) + \sum_{j=1}^{M} K_j v_j$
6. Update the time variable $t = t + \tau$.
7. Repeat the steps 2-6 until $t \leq T_{final}$.

Note: step 3 indicates that τ does not have to be a constant throughout the simulation, which allows for adaptive step leaping algorithms, as we shall discuss later. Adaptive time-stepping is tightly connected with the leap condition and the estimation of the error. The advantages of the tau-leaping method are apparent. We do not have to keep track of all the possible states of the system within a small time interval, but rather trace the dynamics of species populations at various points in time. This saves computational space and gives computational efficiency. If reactions are frequent, we save computational time by simulating them all at once during a leap using the tau-leaping method instead of simulating each one of them as the SSA does. This brings gain in the computational time. It is desired that the total number of reactions that fire over each leap is larger than M, the number of reactions. For that purpose τ for each leap should be larger than a small multiple of $1/a_0(x)$, which is evaluated before the leap. Otherwise, the leaping algorithm is less efficient than the SSA and we lose accuracy without gaining speed. This happens due to the fact that for each leap we have to generate M random numbers. If the sum is of order M, then the system has not moved further in time by much (Tau-Leap algorithm loses efficiency compared to SSA) and we have lost computational time to generate the random numbers. In such a case, it is preferred to abandon the leaping algorithm and use the SSA to advance the state of the system some small number of reactions ahead, and then return to the leaping algorithm.

There are other issues with the Leaping algorithm. Since it is an approximate algorithm, it may lead to negative population numbers. Also, its performance is reduced when simulating stiff systems. We will talk in some detail about these issues.

The Poisson tau-leaping, due to its unrestricted nature, may drive some population sizes to negative numbers. Several factors were suggested as possible culprit and some strategies were offered to deal with the issue (depending on the perceived cause of the negative populations). The paper [19] described a mid-point tau-leaping method and proposes another way to deal with the issue of negative population numbers by using binomially distributed variables instead of the Poisson distributed variables. The mid-point method uses the predicted mid-point (X($t + \tau/2$)) state calculated by considering the mean of the appropriate Poisson variables for each reaction

$$X(t + \tau/2) = x + \frac{\tau}{2} \sum_{j=1}^{M} a_j(x) v_j \ .$$

In this method, the authors used the fact that the binomially distributed variables are bounded (this allows to specify the maximum amount of reactions that fired during the time leap), as well as the fact that the Binomial variable with parameters p and N can be approximated by a Poisson variable with mean $N_j p$, whenever N_j is relatively large and p is relatively small. N_j is estimated for each reaction R_j independently, with regards to the amount of reactant species available. Here p is $a_j \tau / N_j$. The authors show that the Poisson prediction mid-point method can be incorporated into the Binomial tau-leaping method to increase the accuracy for the simulation of the stiff systems. Although it was first suggested that the unbounded nature of the Poisson variable is the culprit [19], it was later observed that more often than not the lack of the coordination between the 'critical' reactions is to blame and that the Binomial leaping approach does not deal with it directly. A critical reaction is a reaction that within a small number of firings can deplete a population of some species. Thus, instead of treating all reactions as equivalent when advancing the system in time, the new approach proposed that the reaction channels should be divided into two mutually exclusive sets: critical and non-critical. In this set-up, a critical moment arises whenever the system is n_c (some user-supplied coefficient, a relatively small integer) number of reactions away from depleting the population of a particular species. Given this notion, we are advancing the system from one critical moment to another (simulating the dynamics of non-critical species between critical reactions using the conventional Poisson tau-leaping) with an

algorithm analogous to the SSA. Whenever we have to simulate the dynamics of the slow reactions, we assume that the sub-system of the fast reactions is in the so-called partial equilibrium. Thus, it is safe to assume that only the average values of the fast-scale reaction propensities affect the dynamics of the slow reactions. This is the approach adopted by the slow scale stochastic simulation algorithm (ssSSA)[3].

One class of systems that needs special attention are the stiff systems. Stiff systems are those with varying time scales of the reaction rates, the fastest of which are stable. To simulate such systems accurately, one should implement the Leap condition that will be influenced by the fastest reaction rate, which means that the Leaping step, τ, will be too small. Therefore the tau-leaping algorithm will not perform efficiently. One method to overcome this problem is to use implicit numerical approximation methods rather than explicit ones. Another approach uses a variable step size algorithm throughout the simulation. This particular family of methods will be the focus of this thesis and we shall introduce an algorithm for it.

Systems with high rates of reactions of each type firing is another important class of problems. An extension of the classical tau-leaping algorithm is possible for them. For such systems not only we should bound τ from being too large, we would also like to have a criterion for τ not being too small. When the populations of all species are large, all of the propensities become relatively large.

Therefore, the time step τ which depends on the propensities will be very small, which is not necessary the case when all reactions' propensities are above a certain threshold. If we will restrict the Leap condition further and bound τ not only from above, but also from below, we will derive another model, the Chemical Langevin Equation, that will give an accurate description of the system's dynamics for this class of systems.

2.4 Chemical Langevin equation

Extended leap condition The time step τ should be small enough so that no propensity function changes significantly over the leap, but it should be sufficiently large so that $a_j(x)\tau >> 1, \forall j$. The second part of the extended Leap condition is true for systems where the populations of all types of species are large, since the propensity functions are proportional to them. By imposing this condition, we reduce the chance that the leaping algorithm does not perform efficiently, since over each leap, we can simulate a number of reactions firing that is larger than the order O(M). It is known that the Poisson random

variable with large mean can be well approximated by the normal distributed random variable with the same mean and variance. Thus, the number of times each reaction channel fired during $[t, t+\tau)$ can be approximated by:

$$K_j = P(a_j(x)\tau) = N(a_j(x)\tau, a_j(x)\tau), \forall j = 1 \dots M \tag{2.20}$$

Note that we can write $N(a_j(x)\tau, a_j(x)\tau) = a_j(x)\tau + \sqrt{a_j(x)\tau}N_j(0,1)$, where $N_j(0,1), \forall j = 1 \dots n$ are independent normally distributed random variables with mean 0 and variance 1.

Substituting (2.20) in the (2.19) Tau-leaping equation leads to the Langevin leaping equation:

$$X(t+\tau) = X(t) + \sum_{j=1}^{M} v_j a_j(X(t))\tau + \sum_{j=1}^{M} v_j \sqrt{a_j(X(t))\tau} N_j(0,1) \tag{2.21}$$

By approximating the Poisson distributed random variables with the normal random variables we moved from integer values for the number of molecules to real values for them. The errors of such an approximation are small when populations of molecules are large. If we subtract X(t) from both sides of the equation, divide both sides by τ and take the limit $\tau \to 0$, we obtain the following equation:

$$\frac{dX}{\mathrm{d}t} = \sum_{j=1}^{M} a_j(X(t))v_j + \sum_{j=1}^{M} v_j \sqrt{a_j(X(t))}\Gamma_j \ , \tag{2.22}$$

where Γ_j are independent Gaussian white noise processes for $1 \leq j \leq M$. The stochastic differential equation (SDE) (2.22) is called the Chemical Langevin Equation (CLE). Unlike the Chemical Master Equation which is a discrete stochastic model, the reduced model of the Chemical Langevin Equation is continuous and stochastic. It is of dimension equal to the number of different reacting species in the system.

There are many numerical methods for approximating the solution of stochastic differential equations. For more details we suggest as reading [15].

In this thesis, to approximate numerically the solution of stochastic differential equations, we use the Euler-Maruyama scheme. This method is of strong order of convergence 1/2 and weak order of convergence 1. Before we proceed with the method, we introduce some definitions.

2.4.1 Brownian motion

Definition 1. *A standard Brownian motion process W(t) over the time period [0,T] is a continuous in time t random variable with the following properties:*

1. *W(0) = 0.*
2. *For any $0 \leq s < t \leq T$, the random variable Z = W(t) - W(s) has a normal distribution with mean 0 and variance t-s.*
3. *For any $0 \leq s < t < u < v \leq T$, the random variables $Z_1 = W(t) - W(s)$ and $Z_2 = W(v) - W(u)$ are independently distributed.*

To simulate the evolution in time of a Brownian motion, we discretized the time interval $[0, T]$ and evaluate the Brownian motion at the mesh points as follows. Assume that the time period $[0, T]$ is divided into L subintervals, each of length Δt, then $\Delta t = T/L$. Thus, $t_i = i\Delta t$ and $W(\Delta t \cdot i) = W(t_i) = W_i$ and $\Delta W_i = W_i - W_{i-1}$. Consequently, $W_i = W_{i-1} + \Delta W_i$, for $1 \leq i \leq L$. However, we know that $\Delta W_i = W(t_i) - W(t_{i-1})$ is a normally distributed random variable with mean 0 and variance $t_i - t_{i-1} = \Delta t$. Hence, we can write $W_i = W_{i-1} + N(0, 1)\sqrt{\Delta t}$, where N(0,1) is a random variable with a normal distribution of mean 0 and variance 1. Thus, in order to simulate a single trajectory over the time period $[0, T]$, with L mesh points, we have to draw L random samples of the Normal distribution random variable with mean 0 and variance 1 and to calculate W_i by: $W_i = W_{i-1} + N(0, 1)\sqrt{\Delta t}$ for $1 \leq i \leq L$.

2.4.2 The Itô integral

Recall the definition of a Riemann integral for a given function h defined on some interval $[0, T]$,

$$\int_0^T h(t)\,\mathrm{d}t \ ,$$

as the limit as $\Delta t \to 0$ of $\sum_{i=1}^{L} (t_{i+1} - t_i)h(\xi_i)$, where $\xi_i \in [t_i, t_{i+1}]$ for any $1 \leq i \leq L$ and $\Delta t = t_{i+1} - t_i$. Stochastic integration gives the integral of a function h with respect to a standard Brownian motion, W(t), and is denoted by

$$\int_0^T h(t)\,\mathrm{d}W(t) \ .$$

Following our analogy with the deterministic integral, the stochastic integral can be approximated by the following sum:

$$\int_0^T h(t)\,\mathrm{d}W(t) = \lim_{\Delta t \to 0} \sum_{i=1}^{N} h(\xi_i)(W(t_{i+1}) - W(t_i)) \ , \tag{2.23}$$

where $\xi_i \in [t_i, t_{i+1}]$. For the stochastic integrals, unlike for the deterministic integral, the choice of ξ_i leads to different values of the integral. This is due to the fact the we integrate with respect to a non-differentiable function. For example ξ_i could be equal to t_i or $\frac{t_{i+1}+t_i}{2}$. The choice of ξ_i produces different results. When $\xi_i = t_i$, the integral (2.23) is called an Itô integral and when $\xi_i = \frac{t_{i+1}+t_i}{2}$ it is called a Stratonovich integral. In this thesis we will use $\xi_i = t_i$, which corresponds to the Itô integral.

2.5 Euler-Maruyama method

A scalar SDE can be written in the following form:

$$X(t) = X(0) + \int_0^t f(X(s))\,\mathrm{d}s + \int_0^t g(X(s))\,\mathrm{d}W(s), t \in [0, T] \ . \tag{2.24}$$

The function f(x) is called the *drift function* and the function g(x) is called the *diffusion function*. The initial condition $X(0) = X_0$ is given. We would like to numerically approximate the solution $X(t)$ of the equation (2.24). The integral form (2.24) can be re-written in differential form as:

$$dX = f(X(t))dt + g(X(t))dW(t) \ , \tag{2.25}$$

where $X(0) = X_0$. To solve the equation numerically we once again have to discretize the time interval and to approximate the solution X(t) at the mesh points. We take $\Delta t = T/L$, where L is the number of intervals. The time variable at the mesh points is $\tau_j = j\Delta t$ for $j = 1, .., L$ and we wish to find the corresponding approximation of the variable $X_j \approx X(\tau_j)$. The Euler-Maruyama method to approximate the solution to the SDE (2.25) takes the following form:

$$X_j = X_{j-1} + f(X_{j-1})\Delta t + g(X_{j-1})(W(\tau_j) - W(\tau_{j-1})), j = 1, \ldots, L \ . \tag{2.26}$$

Thus, for a single trajectory one needs to generate a Brownian path with L points W_j to estimate $W(\tau_j)$, $j = 1, \ldots, L$ and to estimate X_j based on the information of X_{j-1}. The Chemical Langevin Equation is an autonomous stochastic differential equation driven, in general, by multiple Wiener processes. Recall that:

$$\mathrm{d}X = \sum_{j=1}^{M} a_j(X(t))v_j\mathrm{d}t + \sum_{j=1}^{M} \sqrt{a_j(X(t))}v_j\mathrm{d}W_j \tag{2.27}$$

Here, $\sum_{j=1}^{M} a_j(X(t))v_j$ is the drift coefficient and $\sqrt{a_j(X(t))}v_j$ are the diffusion coefficients. Hence, using the Euler-Maruyama method, we obtain the following algorithm for simulating the approximate solution of the CLE: Given the initial state of the system $X(t_0) = X_0$ at the start time t_0 and the number L of the intervals do:

1. Set $\tau = T/N$
2. Set $t = t_0$ and i = 1.
3. Generate an $M \times L$ array A, with each entry being a sample of the normal distributed random variable with mean 0 and variance 1.
4. Update the state system at time $t+\tau$ by $X(t+\tau) = X(t)+\sum_{j=1}^{M} v_j a_j(X(t))\tau+\sum_{j=1}^{M} v_j\sqrt{a_j(X(t))\tau}A_{ji}$
5. Update the time variable $t \to t + \tau$.
6. Update i = i+1.

Repeat the steps 4-7 until $t \geq T$.

Definition 2. *A method is said to have a* strong order of convergence γ*, if there exists a constant C, such that for* $\Delta t > 0$ *sufficiently small, the following holds true:*

$$E[|X_n - X(n\Delta t)|] \leq C\Delta t^{\gamma}, \forall n \tag{2.28}$$

We are bounding the mean absolute error, or equivalently measuring the rate with which it decays, as $\Delta t > 0$ is sufficiently small. It can be shown, that for appropriate drift and diffusion terms in the SDE the Euler-Maruyama method has strong order of convergence $\gamma = 1/2$. An example of a method of strong order of convergence 1 is the Milstein method.

Definition 3. *A method has* weak order of convergence β *if for all* $2(\beta+1)$ *differentiable functions* q *there exists a constant* C *such that for any* Δt *small enough:*

$$|E[q(X_n)] - E[q(X(n\Delta t))]| < C\Delta t^{\beta}, \forall n. \tag{2.29}$$

It can be shown that provided that the drift and the diffusion terms satisfy certain conditions, the Euler-Maruyama method has a weak order of convergence of order 1. The reason for this is the difference between the Itô calculus and the deterministic calculus, which in particular influences the Taylor expansion.

2.6 Reaction rate equations (RRE)

We discuss in this section the connection between the stochastic mathematical models of biochemical systems presented so far and the traditional reaction rate equation model of biochemical kinetics.

For these systems we assume the thermo-dynamic limit is approached, that is the volume $\Omega \to \infty$ and all species population grow to ∞, such that the concentration X_i/Ω remains constant for all $i = 1, \ldots, N$. Then, it was shown that the propensity functions grow linearly with respect to the system size. This is obvious for unimolecular reactions. The same phenomenon is observed for bimolecular reactions, since their reaction rates are inversely proportional to the volume size. The larger the volume is, the lower is the chance of such a reaction to take place. Now, since we assumed that all species populations are very large, we can use the Chemical Langevin Equation model. As propensities grow linearly, also the drift term grows linearly. However, the diffusion terms (which are the square root of propensities) grow like the square root of the system size and thus becomes negligibly small compared to the drift term. Thus, under the conditions above, the Chemical Langevin Equation may be approximated by:

$$\frac{dX}{dt} = \sum_{j=1}^{M} a_j(X(t))v_j \tag{2.30}$$

The equation (2.30) is known as the Reaction Rate Equation model. It is a system of ordinary differential equations of dimension equal to the number of reacting species in the system.

The connections between the modeling approaches for isothermal well-stirred biochemical systems we introduced so far, are presented in Figure 2.1.

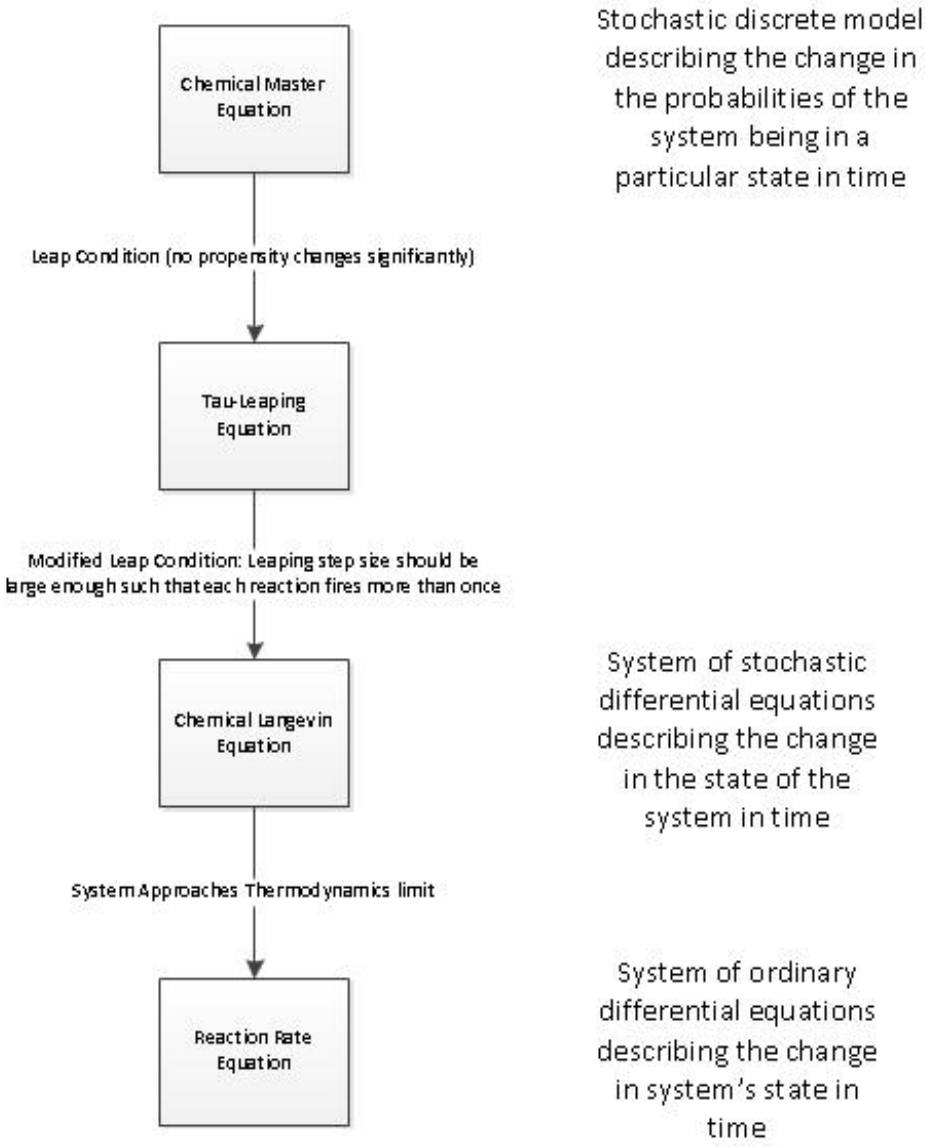

Figure 2.1: The mathematical models of well-stirred biochemical systems

2.7 Implicit methods for the numerical approximation of stiff systems

In order to produce numerical approximations of stiff systems with explicit methods, the numerical method would have to advance in time with steps of the same order as the fastest rate in the system. This leads to very small time steps that render the approach of the leaping algorithms inefficient. A

widely used approach to simulating stiff systems is to employ so-called implicit methods. For example, to solve numerically the following system:

$$\frac{dy}{dt} = f(t, y) \tag{2.31}$$

$$y(0) = y_0 \tag{2.32}$$

with the implicit Euler scheme, we apply the following reccurence:

$$Y(t + \Delta t) = Y(t) + f(t + \Delta t, Y(t + \Delta t))\Delta t \tag{2.33}$$

The equation (2.33) can be solved for the unknown $Y(t + \Delta t)$ by a number of iterative (if the system is non-linear) numerical methods, such as the Newton's method. By using implicit methods we would be able to take larger time steps, as the time steps will not be restricted by the same stability restrictions imposed by explicit methods. This outweighs the extra iterations we will need to do for solving the implicit equation for $Y(t + \Delta t)$. For stochastic models this problem becomes more complicated. If the system has reached the equilibrium with respect to the fast reactions, then the random fluctuations disturb this equilibrium. If these fluctuations are relatively small, then the system will return to the state of slow equilibrium relatively fast and the implicit algorithm may take large steps (larger than the time scale of the fast reactions). In the opposite case, the implicit algorithm due to its taking larger steps than the time scale of the fast reactions will produce the dampening of the fast reaction fluctuations. This can be corrected by down-shifting [5]. The slow reactions will be computed with their correct distributions.

Another issue with the implicit algorithms for stochastic differential equations is the fact that X(t) is a random variable, i.e. it is not known how to set up a scheme which is implicit in both the drift and the diffusion terms. Taking the case of the tau-leaping, this issue is dealt with as follows, the mean of Poisson variable, $a_j(x)\tau, \forall j = 1, .., M$, is approximated implicitly, that is at $t + \tau$ (deterministic part), and the remaining stochastic part is estimated explicitly at t. The corresponding equation is:

$$X(t + \tau) \approx X(t) + \sum_{j=1}^{M} v_j a_j(X(t + \tau))\tau + \sum_{j=1}^{M} v_j(P_j(a_j(X(t)), \tau) - a_j(X(t))\tau) \tag{2.34}$$

Once again, we obtained an implicit equation in $X(t+\tau)$, which can be solved by using an iterative method. The corresponding implicit algorithm for the Chemical Langevin equation is:

$$X(t+\tau) \approx X(t) + \sum_{j=1}^{M} v_j a_j(X(t+\tau))\tau + \sum_{j=1}^{M} v_j(\sqrt{a_j(X(t))\tau}N_j(0,1)) \tag{2.35}$$

Here the drift term is estimated at $t+\tau$ and the diffusion term is evaluated at t. As before, $N_j(0,1)$ are independent normally distributed random variables with mean 0 and variance 1.

2.8 Slow scale SSA for stiff systems

Another method for simulating stiff biochemically reacting systems is to employ the slow scale SSA (ssSSA)([3]). It differs in principle from the implicit methods described above. It has some resemblance with the hybrid methods using both the SSA and the leaping algorithms. At the heart of ssSSA lie two assumptions. First being that there are at least two time scales in the system that are drastically different from each other. The second being that the fast reactions reach an equilibrium (or pseudo-equilibrium) relatively fast with respect to the time scale of the slow reactions. By pseudo-equilibrium we mean that the chemical reactions might still take place, but the quantities of all the species participating in the slow reactions remain relatively constant.

We partition the space of all species into two subspaces: the slow species and the fast species. Fast species are species whose quantity is being changed by some fast reaction. Slow species are the ones that are not fast. Thus, a fast species may be changed by the slow reaction, but a slow species can not be changed by a fast reaction. Fast reactions have propensity functions that are relatively large in comparison to the propensities of the slow reactions. Note that the propensities are dependent on the number of reactants present in the system as well as on the rate of the corresponding reaction. We denote the fast reactions by $R^f = [R_1^f, .., R_{M^f}^f]$ and the slow reactions by $R^s = [R_1^s, .., R_{M^s}^s]$. Here, M^s is the number of fast reactions and M^s is the number of slow reactions and $M^f + M^s = M$. Likewise, we partition the space of all species into two subspaces. We denote the fast species by $S^f = [S_1^f, .., S_{N^f}^f]$ and the slow species by $S^s = [S_1^s, .., S_{N^s}^s]$. Here, N^s is the number of fast species and N^s is the number of slow species. Therefore, $N^f + N^s = N$. Since we partitioned the space of species into two subspaces, the same can be done for the state vector, $X(t) = (X^s(t), X^f(t))$. Bearing in mind the lack of symmetry

between the fast and the slow species we divide the state change vectors onto v_j^f and v_j^s. Here:

$$v^f = [v_{1j}^{ff}, ..., v_{N^f j}^{ff}]^T, j = 1, .., M^f \tag{2.36}$$

$$v^s = [v_{1j}^{ss}, ..., v_{N^s j}^{ss}, v_{1j}^{fs}, ..., v_{N^f j}^{fs}]^T, j = 1, .., M^f \tag{2.37}$$

Here, $v_{i,j}^{\alpha\beta}$ denotes the change brought by one reaction R_j^β of either slow or fast type (denoted by β) in the slow species i of either the slow or the fast subspace (denoted by α).

Having set up a system in this way, we focus on the system that consists only of the fast reactions of the initial system. We denote it by $\hat{X}^f(t)$. For the ssSSA to be applicable, the following conditions should be true. The process $\hat{X}^f(t)$ should be stable, which means the following limit exists:

$$\lim_{t\to\infty} \hat{P}(x^f, t|x_0, t_0) = \hat{P}(x^f, \infty|x_0) \ , \tag{2.38}$$

where $\hat{P}(x^f, t|x_0, t_0)$ is the conditional probability for the system of the fast reactions only. The second condition is that $\hat{X}(t)^f$ reaches its limit value $\hat{X}(\infty)^f$ very quickly (compared to the time scale of the slow reactions). Having this set-up and assuming that by the time a slow reactions fires, the fast reaction system is at equilibrium, we can estimate the probability that a fast reaction fires during time interval $(t; t + dt_s]$, $a_j^s dt_s$ using:

$$\bar{a}_j^s = \sum_{x^{f'}} \hat{P}(x^{f'}, \infty|x^f, x^s) a_j^s(x^{f'}, x^s) \ , \tag{2.39}$$

The state of the system at time t is $X(t) = (x^f, x^s)$. Here $x^{f'}$ stands for the state of the system with slow reactions switched off. Thus, $\bar{a}_j^s$ the slow scale propensity function averaged over the fast variables, as if these variables had distribution with the slow reactions turned off.

The algorithm obtained from this set-up leads to the system moving from one slow reaction to the next, while updating the fast species by randomly sampling the virtual fast process $\hat{X}^f(t)$ according to (2.38).

2.9 Reflection on the stepsize

For the Chemical Langevin Equation model to apply there are two limitations on the step size τ. The first condition is the Leap condition and the second is that the step size should be large enough so that many reactions fire during one step. We need a way to determine the largest τ so that both conditions hold.

One way to bind the step size τ is to ensure that the change in the state of the system will not bring a change in any propensity function larger than the sum of all propensities [10]. In other words, is we assume that $X(t) = x$ and $X(t+\tau) - X(t) = \Phi(\tau, x)$, then

$$|a_j(x + \Phi(\tau, x)) - a_j(x)| \leq \epsilon a_0(x), \forall j = 1, .., M \ . \tag{2.40}$$

Here $0 < \epsilon << 1$ and $a_0(x) = \sum_{j=1}^{M} a_j(x)$. Improved ways of bounding the change in each propensity have been proposed in [2], [4]. Each propensity function $a_j(x)$ may be bounded in terms of a small fraction of itself or in terms of a fraction of the largest propensity function at time t. Mathematically, this leads to the following conditions:

$$|a_j(x + \Phi(\tau, x)) - a_j(x)| \leq \epsilon a_j(x), \forall j = 1, .., M \ , \tag{2.41}$$

or

$$|a_j(x + \Phi(\tau, x)) - a_j(x)| \leq \epsilon \max_{i=1..N}[a_i(x)], \forall j = 1, .., M \ . \tag{2.42}$$

The task of choosing the appropriate leaping step τ to satisfy any of the above three conditions is being complicated by the fact that $\Phi(\tau, x)$ is a random variable and we have to draw many representative samples in order to accurately determine all possible outcomes of using the leaping step τ of a particular size.

Chapter 3

Method

In this section, we propose an adaptive time-stepping algorithm for the numerical solution of the Chemical Langevin Equation. We assume that the extended leaping condition holds. Some systems exhibit a variable behavior throughout the simulation: they might be stiff during one period of time and non-stiff on another. It is not efficient then to employ the same small step size to ensure that the error is below some pre-fixed tolerance. Since we compare the error to some tolerance, there may be occasional rejections of the step size. Then the step is substituted by a smaller one (which should lead to a smaller error). However, since this algorithm is based on an a priori-fixed Brownian path, we would have to discard some random variables associated with the rejected step size. This will result in a skew distribution and thus will lead to a less accurate simulation. To avoid this, we will maintain the correct statistics by conditioning the random variables for the new, smaller step on the already generated values. We use the Euler-Maruyama method which is shown to be p-th mean convergent [17].

Another aspect of the proposed algorithm is the way in which the trajectory generation is connected to the error estimation. The stochastic simulation by Monte Carlo type methods involves producing a large number of trajectories, which may be used to approximate the statistics for the exact solution of the problem. These trajectories are computed one at a time. However, if the algorithm uses an adaptive step size scheme, then one needs to verify if the numerical solution converges to the exact solution. The existing convergence theory for adaptive methods for SDEs is developed for the strong convergent schemes.

In the current algorithm, the trajectories are computed synchronously, one time leap at a time. Note

that the number of mesh points in each trajectory will be the same. Thus, if the time step is rejected, it is rejected for all trajectories. This approach has the following advantage. It allows a single error estimation per all trajectories per time step, based on the mean across all generated paths. To compare, generating one trajectory at a time makes it is necessary to evaluate the errors for each time step on each trajectory.

The algorithm we propose is based on the estimate of the p-th mean of the local error (at each step). Moreover, we only compute the leading term in the p-th mean error term. This enables us to estimate the global error, provided that the system has small noise. According to [17], we define the p-mean error of an integrable random variable x by :

$$||x||_{L_p} = (E(|x|)^p)^{1/p} \tag{3.1}$$

Definition 4. *Given the discretized solution at the mesh points, X_l approximates $X(t_l)$, we call an approximation method* p-mean convergent *of order $\gamma > 0$ if*

$$\max_{l=1..N} ||X(t_l) - X_l||_{L_p} \leq C \cdot h^{\gamma} \ , \tag{3.2}$$

where $h = \max_{i=1..L}(h_i)$ and C is some positive constant, independent of h.

According to [17], the p-th mean of the global error term $(X(t_l) - X_l)$ can be estimated by a term proportional to p-th mean of a local error. We discuss below two important aspects to this strategy: the error estimation and the time-stepping technique.

Consider the following stochastic differential equation:

$$X(t) = X(0) + \int_0^t f(X(s))\,\mathrm{d}s + \int_0^t g(X(s))\,\mathrm{d}W(s), t \in [0, T] \ , \tag{3.3}$$

where $W = [W_1, W_2, .., W_M]^T$ is an M-dimensional Wiener process, having independent entries. We use the following results from [17] to estimate the leading term of the local approximation error (η_l) for the Euler-Maruyama scheme applied to problems with small noise:

$$\eta_l = h\frac{1}{2}||(f_t + f_x f)(x_{l-1}, t_{l-1})||_{L_p} \ , \tag{3.4}$$

where, h is the time step.

In our case, the stochastic differential system is the Chemical Langevin Equation:

$$X(t) = X(0) + \int_0^t \sum_{j=1}^{M} a_j(X(s))v_j \, \mathrm{d}s + \int_0^t \sum_{j=1}^{M} \sqrt{a_j(X(s))}v_j \, \mathrm{d}W_j(s) \ , t \in [0, T] \tag{3.5}$$

Thus, the drift coefficient is $f = \sum_{j=1}^{M} a_j(X(t))v_j$. We remember the leaping assumption that $a_j, \forall j = 1, .., M$, remains almost constant during the leap. Therefore, $f_t = 0$. The approximate leading error term can be estimated as following:

$$\eta_l = h_l \frac{1}{2} ||(f_x f)(x_{l-1}, t_{l-1})||_{L_p} \ . \tag{3.6}$$

3.1 Adaptive time-stepping method

There are two aspects to the proposed adaptive algorithm: the error evaluation in the perceived post-leap state of the system and the correction of the time step accordingly. The key idea of this algorithm is to be able to maximize the mean square error of the approximation, while satisfying the prescribed tolerance. Thus, we strive for the error being exactly equal to the tolerance. This not only allows us to diminish the step size when the error exceeds the user-provided tolerance, but also increases the step size, when the error falls way below it (and thus prevents unnecessary increase of computational time).

The algorithm we employ for numerically solving the Chemical Langevin Equation is the Euler-Maruyama method described above. We propose below an adaptive time-stepping algorithm for the Euler-Maruyama scheme applied to the CLE. We will show that this leads to a numerical solution which converges to the exact solution of the given problem.

3.2 Time step adjustment

Before attempting a step we calculate the mean square error across all the trajectories, η, by using the current state of the system and the current value of τ. Since we would like for the step size to decrease only when the error for the current time step is larger than the provided tolerance, we would like to make the step size adjustment a function of the ratio between the local error and the tolerance. Based on the resulting error, we calculate the candidate time step for the next iteration using the following

equation:

$$\tau_{new} = \tau_{old} \cdot \left(\frac{\theta \cdot tol}{\eta} \right)^{1/\gamma} , \tag{3.7}$$

and $|\theta| < 1$ is a safety factor. If for the current time step τ the error is smaller than the tolerance, then we accept the step, calculate the number of times each reaction fired during the time step, update the system correspondingly, advance the time by τ and re-set τ. This strategy is particularly useful when the numerical solution of the system is close to equilibrium and no sharp changes in any species population is expected.

If the error associated with the current step is higher than the tolerance, we divide the current time step τ into smaller steps $\tau_1 = \tau_{new}$ and $\tau_2 = \tau - \tau_1$. However, previously we had a set of normal random variables associated with the current τ. Since we wish to maintain the correct distribution of the numerical solution, we use these random variables for the calculation of the number of reactions that fired during τ_1 and τ_2. If we denote the Wiener increment on the larger interval $[t; t + \tau]$ by ΔW, then ΔW has mean 0 and variance τ. Then, the increments ΔW_1 and ΔW_2 corresponding to $[t, t + \tau_1)$ and $[t + \tau_1, t + \tau)$ are calculated according to [17], by the following two equations:

$$\Delta W_1 = \frac{\tau_1}{\tau} \Delta W + \sqrt{\frac{\tau_1 \tau_2}{\tau}} \phi \tag{3.8}$$

$$\Delta W_2 = \frac{\tau_2}{\tau} \Delta W - \sqrt{\frac{\tau_1 \tau_2}{\tau}} \phi \, , \tag{3.9}$$

where ϕ is a vector of length M, each entry of which is $N(0, 1)$. One notices that $\Delta W_1 + \Delta W_2 = \Delta W$. The important consideration for the stepping strategy is that we try to keep the distribution un-altered, which means that when a time step was rejected, and a smaller time step was chosen, then we need to condition the Wiener increments on the smaller time steps on the Wiener increments associated with the rejected step, by using (3.8) and (3.9).

3.3 Algorithm

We introduce below an algorithm for adapting the step size for the mean-square numerical approximation of the solution of the Chemical Langevin Equation. This algorithm applies when the noise in the

Chemical Langevin Equation model is relatively small. This is the case for a large class of problems arising in applications.

Input:

1. Initialize t = 0 and the initial state of the system is $X(t) = X_0$. Set i=1, set the number of trajectories, the tolerance tol, the control factor $\theta = 0.8$ and the initial step τ^1.

2. For all the trajectories generate the vectors $\Delta W = (\Delta W_1, ..., \Delta W_M)^T$ each having M independent Gaussian random variables as entries, with mean 0 and variance the current time step size, τ.

3. Based on the current state of the system and the current value of the time step, we calculate the mean-square error η_i across all trajectories, according to (3.6) where the drift is

$$f(x) = \sum_{j=1}^{M} a_j(x) v_j \ .$$

4. Calculate τ_{new} using (3.7).

5. Compare the error for the current step with the tolerance. If the error η_l is smaller than the tolerance tol then accept the step $t_i = t_{i-1} + \tau^i$:

   (a) Update the system state variable using the Euler-Maruyama scheme:

$$X_i = X_{i-1} + \sum_{j=1}^{M} a_j(X_{i-1}) v_j \tau^i + \sum_{j=1}^{M} \sqrt{a_j(X_{i-1})} v_j \Delta W_j$$

   (b) If $t_i < T_{final}$, set i = i+1, $\tau^i = \tau_{new}$ and return to Step 3. Otherwise, stop.

   Else:

   (a) Set $\tau_1 = \tau_{new}$ and $\tau_2 = \tau^i - \tau_1$.

   (b) Generate the Wiener increment vectors for τ_1 and τ_2 across all trajectories according to (3.8) and (3.9).

   (c) Update the system state variable X_i, using the Euler-Maruyama scheme and set i=i+1

   (d) Set $\tau^i = \tau_2$.

   (e) Go to step 3.

The following theorems apply, as described in [17].

3.4 Convergence analysis of the method

Definition 5. *A function F:* $\Omega \subset R^n \to R^n$ *is called* Lipschitz continuous, *if for all* X_1 *and* $X_2 \in \Omega, \exists K$ *constant, such that:*

$$|F(X_1) - F(X_2)| \leq K|X_1 - X_2| \tag{3.10}$$

Effectively, this limits the absolute value of rate of growth of the function (i.e. the absolute value of its derivative if it exists) to be no larger than a constant K. The constant K is called the Lipschitz constant for F.

The following theorem establishes the p-th mean stability of the described method.

Theorem 1. *Consider the following Itô stochastic differential equation:*

$$X(t) = X_0 + \int_{t_0}^{t} f(x(s), s)\,\mathrm{d}s + \int_{t_0}^{t} g(x(s), s)\,\mathrm{d}W(s), t \in T \tag{3.11}$$

Here, $X_0 = X(t_0)$, $T = [t_0, T_{final}], f : R^n \times T \to R^n, g : R^n \times T \to R^{n\times m}$ *and f is continuously differentiable with respect to t. Let the following scheme be the discretization scheme for the solution of (3.11):*

$$X_k = X_{k-1} + F(X_{k-1}, X_k, t_{k-1}, \tau_k) + G(X_{k-1}, t_{k-1}, \tau_k, I_{t_{k-1},h_k}), k = 1..N \tag{3.12}$$

Here I_{t_{k-1},h_k} *is a vector of multiple stochastic integrals. Also, let* $p \geq 1$ *and* X_0 *to have a p-th order moment. Assume that the following properties hold for the scheme (3.12):*

- *For all the random vectors* X_k^1 *and* X_{k-1}^1, X_k^2 *and* X_{k-1}^2, *k=1,..,N and the pair* $(t, \tau) \in T$, *the following holds:*

$$|F(X_k^1, X_{k-1}^1, t, \tau) - F(X_k^2, X_{k-1}^2, t, \tau)| \leq \tau(L_1|X_k^1 - X_k^2| + L_2|X_{k-1}^1 - X_{k-1}^2|) \tag{3.13}$$

  for some positive constants L_1, L_2 *and* τ.

- *For all the pairs* $(t, \tau) \in T$ *and for all the random vectors* X^1 *and* X^2

$$E(|G(X^1,t,\tau,I_{t,\tau})-G(X^2,t,\tau,I_{t,\tau})|)=0\ .$$

$$E(|G(X^1,t,\tau,I_{t,\tau})-G(X^2,t,\tau,I_{t,\tau})|^p)\le\tau^{p/2}L_3^p|X^1-X^2|^p\ ,$$

for some constant L_3 .

$$E(|G(0,t,\tau,I_{t,\tau})|^p)<\infty\ .$$

Then for all $a\ge 1$ *there exist a maximal step size* $\tau^0>0$ *and a stability constant* $S>0$ *such that the following holds for each set* $t_0,t_1,..,t_N$ *having the property* $\tau=\max_{k=1..N}\tau_k\le\tau_0$ *and* $\tau\times N\le a(t_N-t_0)$*: for all the vectors* $X_0^*,\hat{X}_0$ *having finite p-th mean and the perturbations* $d_l^*,\hat{d}_l$ *having a finite p-th mean* $\forall k=1,..,N$*, the perturbed discrete system*

$$\hat{X}_k=\hat{X}_{k-1}+F(\hat{X}_{k-1},\hat{X}_k,t_{k-1},\tau_k)+G(\hat{X}_{k-1},t_{k-1},\tau_k,I_{t_{k-1},\tau_k})+\hat{d}_k,\ k=1..N \tag{3.14}$$

has a unique solution $(\hat{X}_k)_{k=0}^N$ *and the following estimates are valid for any two solutions* $(\hat{X}_k)_{k=0}^N$ *and* $(X_k^*)_{k=0}^N$ *of the perturbed discrete system with perturbations* $(d_k^*)_{k=1}^N$ *and* $(\hat{d}_k)_{k=1}^N$ *and splittings of* $d_k=d_k^*-\hat{d}_k$ *such that* $d_k=s_k+r_k$ *with* $E(s_l)=0$

$$E(\max_{k=1..N}|X_k^*-\hat{X}_k|^p)\le S^p(E|X_0^*-\hat{X}_0|^p+\frac{\max_{k=1..N}E|s_k|^p}{h^{\frac{p}{2}}}+\frac{E\max_{k=1..N}|r_k|^p}{h^p}) \tag{3.15}$$

$$\max_{k=1..N}E|X_k^*-\hat{X}_k|^p\le S^p(E|X_0^*-\hat{X}_0|^p+\frac{\max_{k=1..N}E|s_k|^p}{h^{\frac{p}{2}}}+\frac{\max_{k=1..N}E|r_k|^p}{h^p}) \tag{3.16}$$

The scheme (3.12) that has properties (3.15) and (3.16) is p-th mean stable.

The following lemma establishes that the drift-implicit Euler scheme applied to (3.11):

$$X_k=X_{k-1}+\tau_k(\alpha f(X_k,t_{k-1}+\tau)+(1-\alpha)f(X_{k-1},t_{k-1}))+g(X_{t-1},t_{k-1})\Delta W_k,k=1,..,N \tag{3.17}$$

is p-th mean stable, where $\alpha \in [0, 1]$. It is easy to see that if we will take

$$F(X, Z, t, \tau) = \tau(\alpha f(X, t + \tau) + (1 - \alpha) f(Z, t)) \tag{3.18}$$

$$G(X, t, \tau, I_{t,\tau}) = g(X, t)(W(t + h) - W(t)) = \sum_{j=1}^{m} g_j(X, t) \int_t^{t+\tau} \mathrm{d}W_j(s) \tag{3.19}$$

the scheme (3.17) turns into scheme (3.11) and thus, if functions f and g satisfy the necessary conditions, the Theorem 1 can be applied to the Euler scheme.

Lemma 1. *Let the functions f and g be Lipschitz continuous with respect to x. Then the Euler scheme (3.17) is p-th mean stable.*

Together Theorem 1 and Lemma 1 establish that the scheme developed in our method (based on Euler numerical method) is p-th mean stable. Note that in this thesis we deal with explicit schemes, that is with the Euler-Maruyama scheme, corresponding to $\alpha = 0$ in (3.17).

Chapter 4

Numerical results

4.1 System 1: Michaelis-Menten model

The first model is well known in the literature as the Michaelis-Menten model, as it was given in [11]. It consists of the following biochemical reactions:

$$\begin{aligned} S_1 + S_2 &\xrightarrow{c_1} S_3 \\ S_3 &\xrightarrow{c_2} S_1 + S_2 \\ S_3 &\xrightarrow{c_3} S_4 + S_2 \end{aligned}$$

This system models the conversion of substrate into product, in the presence of an enzyme. We denoted the substrate by S_1, the enzyme by S_2, the complex substrate-enzyme by S_3 and the product by S_4. The data used is: the parameters are $C = [1.661e-3, 1e-4, 0.1]$. The initial conditions are $X(t_0) = [312, 125, 0, 0]$. We simulated the dynamics of the system for 55 sec using 100 trajectories. The stoichiometric matrix is:

$$V = \begin{pmatrix} -1 & 1 & 0 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

According to Section 2 we obtained the following propensity functions for the first, second and third

	Tol, %	No of Steps	rejected	$\frac{\max error}{tolerance}$
Adaptive	1.14	28	9	0.834
Fixed	1.14	28	0	10.528
Adaptive	0.23	61	0	0.833
Fixed	0.23	61	0	24.162
Adaptive	0.14	113	0	0.880
Fixed	0.14	113	0	26.086
Adaptive	2.28×10^{-2}	517	0	0.856
Fixed	2.28×10^{-2}	517	0	28.508

Table 4.1: System 1: Comparative analysis between the adaptive step and the fixed step algorithms for Michaelis-Menten system

reactions, respectively:

$$\begin{aligned} a_1(X(t)) &= c_1 X_1(t) X_2(t) \\ a_2(X(t)) &= c_2 X_3(t) \\ a_3(X(t)) &= c_3 X_3(t) \end{aligned}$$

Based on the well-studied behavior of this model, we expect our algorithm to perform more efficiently (use less steps and adjust to tolerance better) than the fixed step size algorithm. We compared the performance of the adaptive step-size algorithm with the fixed-step size Langevin Leaping algorithm. We measured the error for the fixed step size algorithm using the same mechanism as for the adaptive algorithm.

In Table 4.1 are summarized the numerical results for the various runs of the adaptive step algorithm and the fixed step size algorithm for the Michaelis-Menten system. We ran the adaptive algorithm for a set of tolerance values, calculated the number of steps obtained while maintaining the error below the tolerance. Then, we ran the fixed step size algorithm for the same number of attempted steps as the adaptive algorithm. At each run, we calculated the ratio between the maximum error and the prescribed tolerance. From table 4.1 we can see that, given the equal number of steps, the error obtained with the adaptive algorithm is between 10 and 28 times smaller than that committed by the fixed step size scheme. Moreover, as the tolerance decreases, the error to tolerance ratio for the fixed step size algorithm grows even more. Figures 4.1-4.4 describe the evolution of the error as a function of time for the two

algorithms.

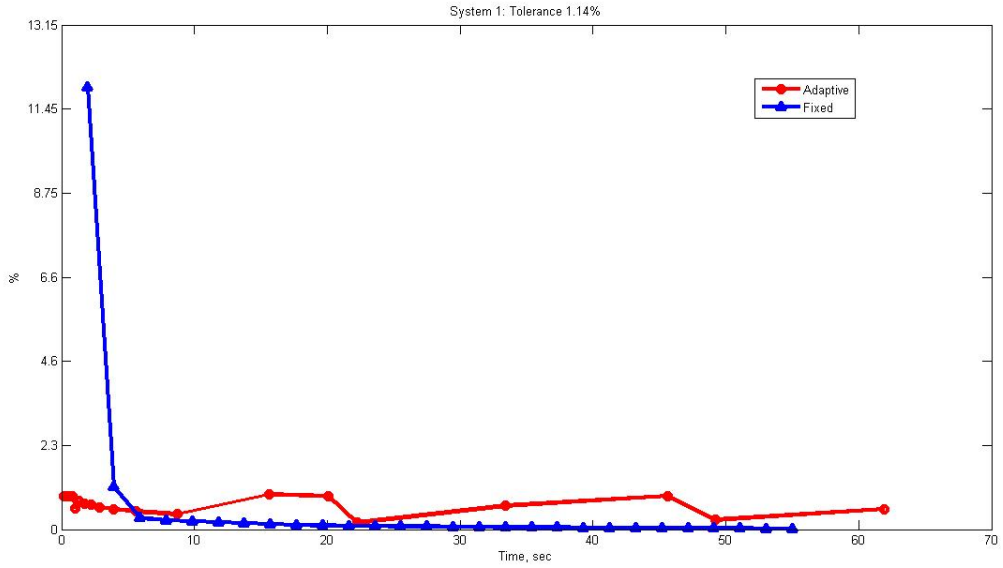

Figure 4.1: System 1, Michaelis-Menten model: plot of the error evolution in time, tolerance = 1.14%

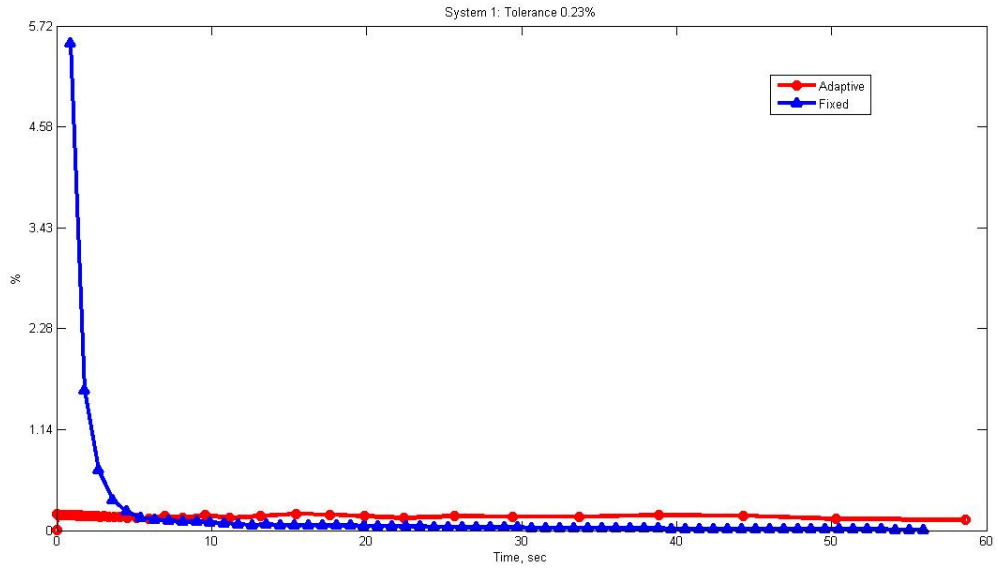

Figure 4.2: System 1, Michaelis-Menten model: plot of the error evolution in time, tolerance = 0.23%

From the graphs we see that the maximum error for the fixed step size method occurs during the

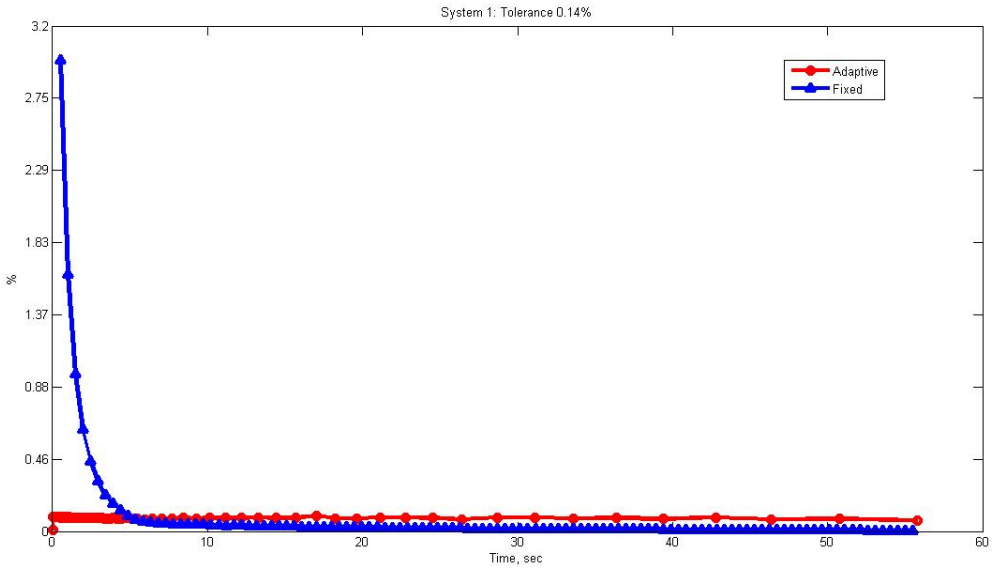

Figure 4.3: System 1, Michaelis-Menten model: plot of the error evolution in time, tolerance = 0.14%

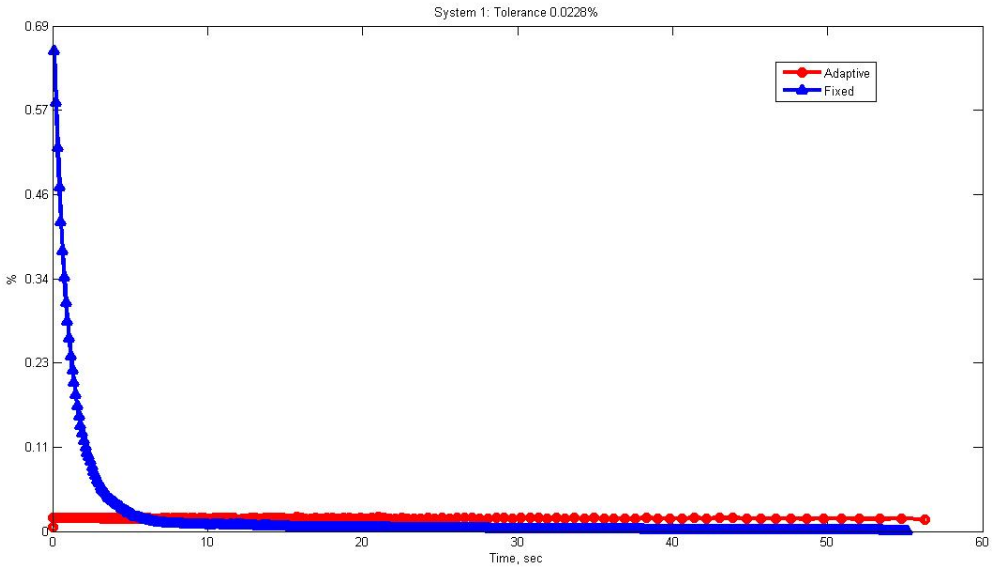

Figure 4.4: System 1, Michaelis-Menten model: plot of the error evolution in time, tolerance = 2.28×10^{-2} %

first 10 sec of the simulation. This happens when the number of molecules has a sharp change. Thus, to have the lower error the step size for the fixed step mesh should have been smaller. However, having an

even smaller step size would reduce the efficiency even further during the time period when the system is not stiff.

Figures 4.5-4.6 plot the system dynamics as estimated with the adaptive and the fixed step size algorithm:

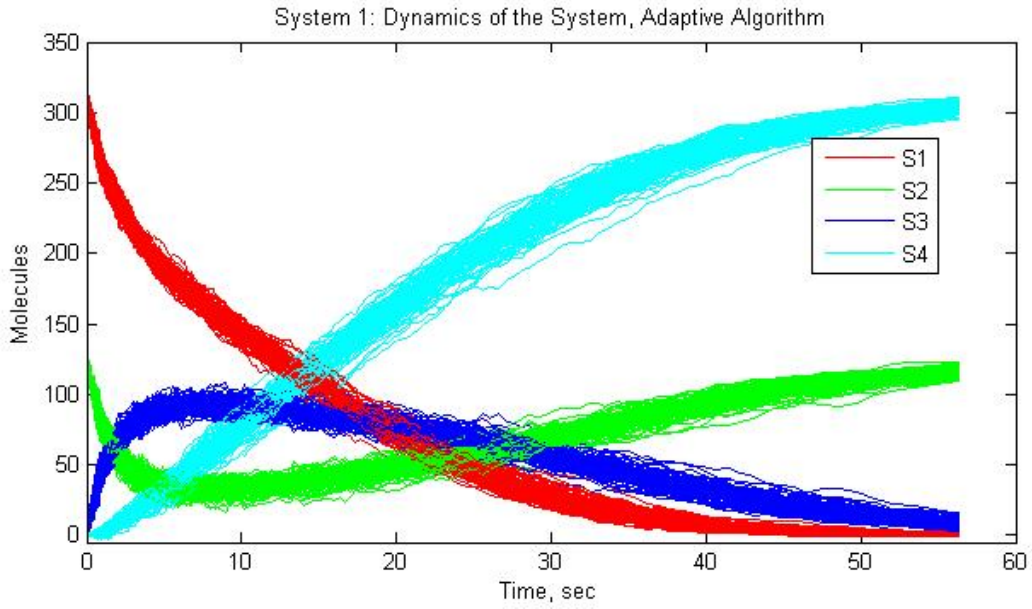

Figure 4.5: System 1, Michaelis-Menten model: plot of system's dynamics, adaptive step algorithm

From Figures 4.5 and 4.6 we see that the adaptive step size algorithm produces a simulation of the system's behavior similar to the one computed by the fixed step strategy. This shows numerically that the adaptive step algorithm gives a good approximation of the exact solution of the problem (it is convergent).

4.2 System 2: decaying dimerization model

The second test model is a stiff decaying dimerization model (see [5] for details). We have modified it slightly, to make the system less stiff. Consequently, the tolerance values could be less strict and consequently less computational time will be required to simulate the evolution in time of the system. The chemical system is subject to the following reactions:

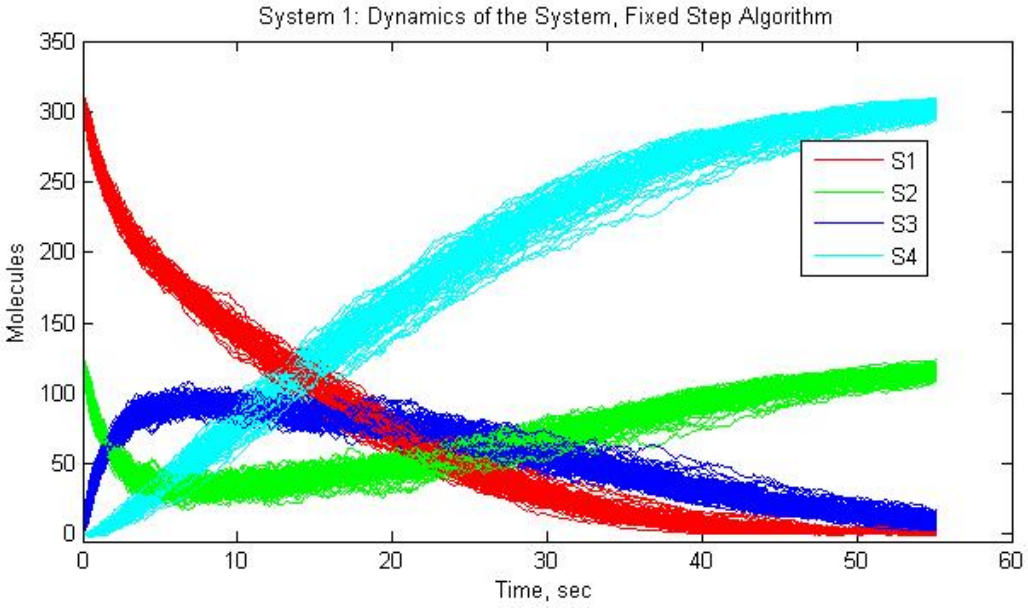

Figure 4.6: System 1, Michaelis-Menten model: plot of system's dynamics, fixed step algorithm

$$
\begin{aligned}
S_1 &\xrightarrow{c_1} \emptyset \\
S_1 + S_1 &\xrightarrow{c_2} S_2 \\
S_2 &\xrightarrow{c_3} S_1 + S_1 \\
S_2 &\xrightarrow{c_4} S_3
\end{aligned}
$$

We use the following data: the parameters are $C = [1, 0.002, 0.5, 0.04]$, while the initial conditions are $X(t_0) = [5000, 10, 10]$. The interval of simulation is $[0, 0.03]$ and we run the algorithm for 100 trajectories. The stoichiometric matrix is:

$$
V = \begin{pmatrix} -1 & -2 & 2 & 0 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

	Tol, %	No of Steps	No of steps rejected	$\frac{\max error}{tolerance}$
Adaptive	10	54	15	0.8
Fixed	10	54	0	1.452
Adaptive	5	75	1	0.8
Fixed	5	75	0	2.091
Adaptive	2.5	150	5	0.8
Fixed	2.5	150	0	2.091
Adaptive	1	373	17	0.8
Fixed	1	373	0	2.102

Table 4.2: System 2: comparative analysis between the adaptive step and the fixed step algorithms for the decaying dimerization model

The propensities corresponding to the decay-dimerization system are:

$$\begin{aligned} a_1(X(t)) &= c_1 X_1(t) \\ a_2(X(t)) &= \tfrac{1}{2} c_2 X_1(t)(X_1(t)-1) \\ a_3(X(t)) &= c_3 X_2(t) \\ a_4(X(t)) &= c_4 X_2(t) \end{aligned}$$

We compared the performance of the adaptive step-size algorithm with that of the regular fixed-step size Langevin Leaping algorithm. Again, we used the same measure for the local error for both the adaptive and the fixed step size algorithm. The results of the simulation with the adaptive step algorithm and the fixed step algorithms are presented in Table 4.2

Similar to the first system, the basic idea of comparison stayed the same. From the table, one can see that given the equal number of total steps (which in the case of the adaptive algorithm means the total number of accepted and rejected steps), the error ratio for the adaptive step algorithm is smaller than that of the fixed step size method. Moreover, as the tolerance size decreases, the error to tolerance ratio for the fixed step size method grows even more compared to the adaptive strategy.

Figures 4.7-4.10 describe the evolution of the errors for the two algorithms, in time:

From these graphs we see that the maximum error for the fixed step size occurs during the first 0.05 sec of the simulation. Thus, to have the lower error the step size for the constant step algorithm should have been smaller. However, this would lead to a decrease in the efficiency during the time period when

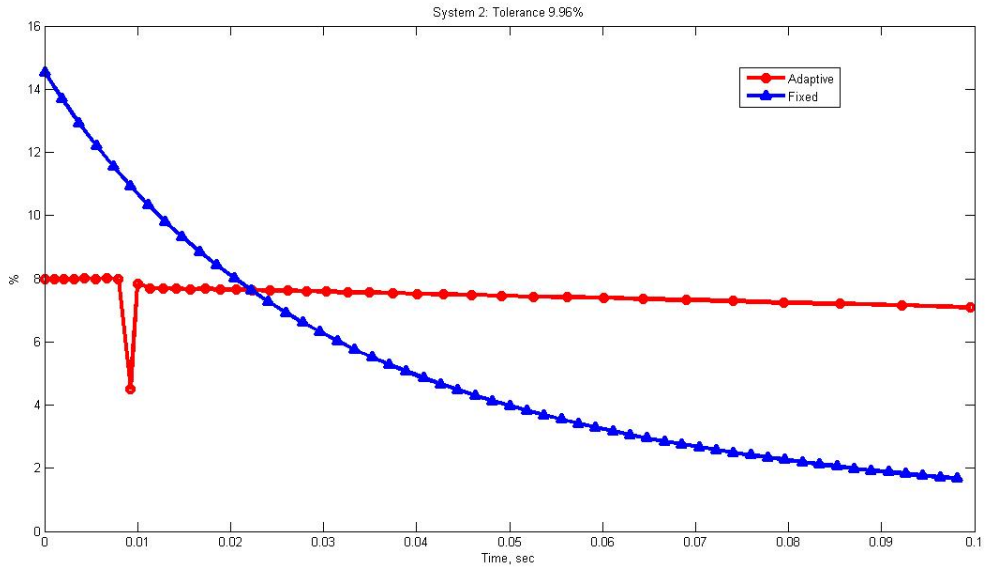

Figure 4.7: System 2, decaying dimerization: plot of the error evolution in time, tolerance = 10%

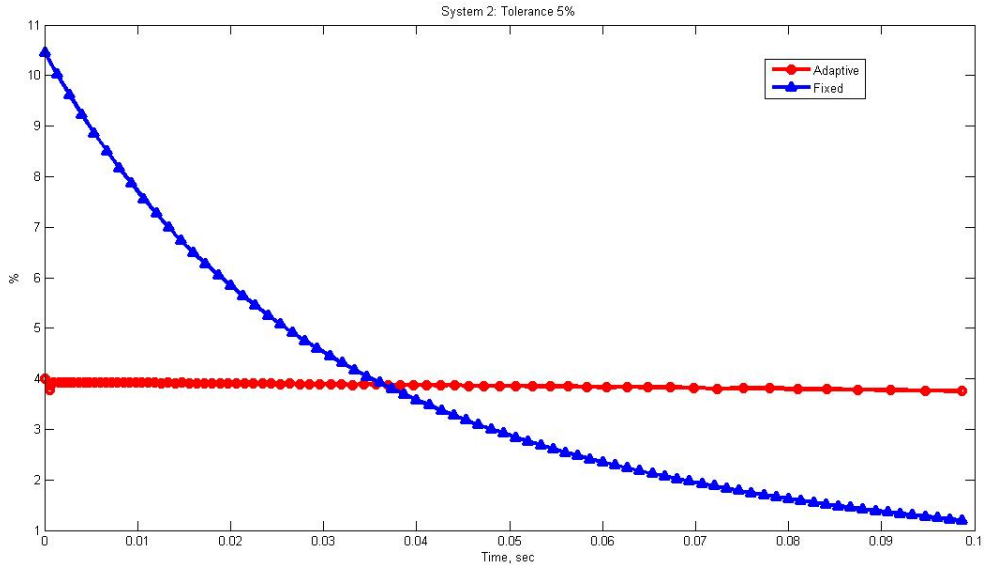

Figure 4.8: System 2, decaying dimerization: plot of the error evolution in time, tolerance = 5%

the system is not stiff. Which in this case, when the system moves away from the dharp transient, the fixed step size algorithm becomes utterly inefficient. The simulation of this system by these two

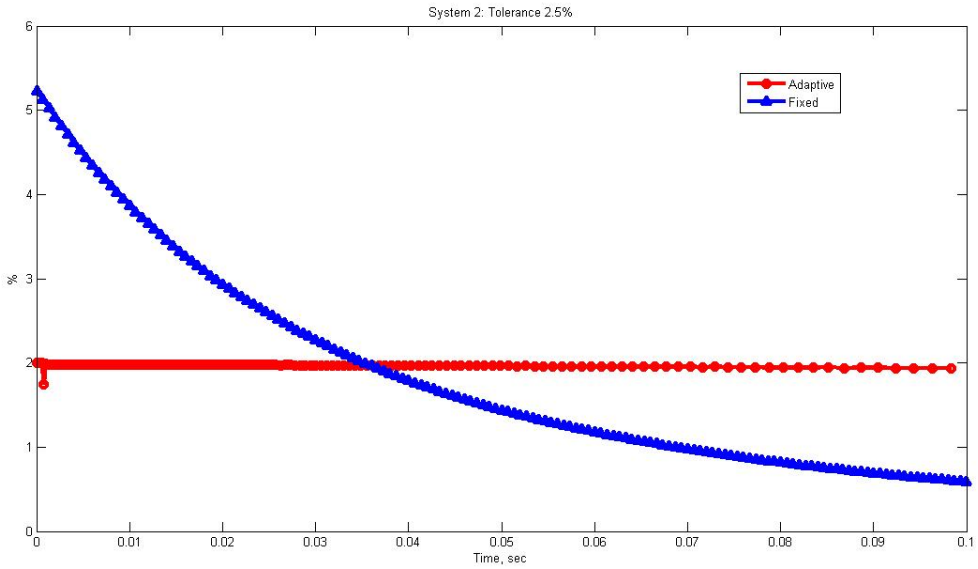

Figure 4.9: System 2, decaying dimerization: plot of the error evolution in time, tolerance = 2.5%

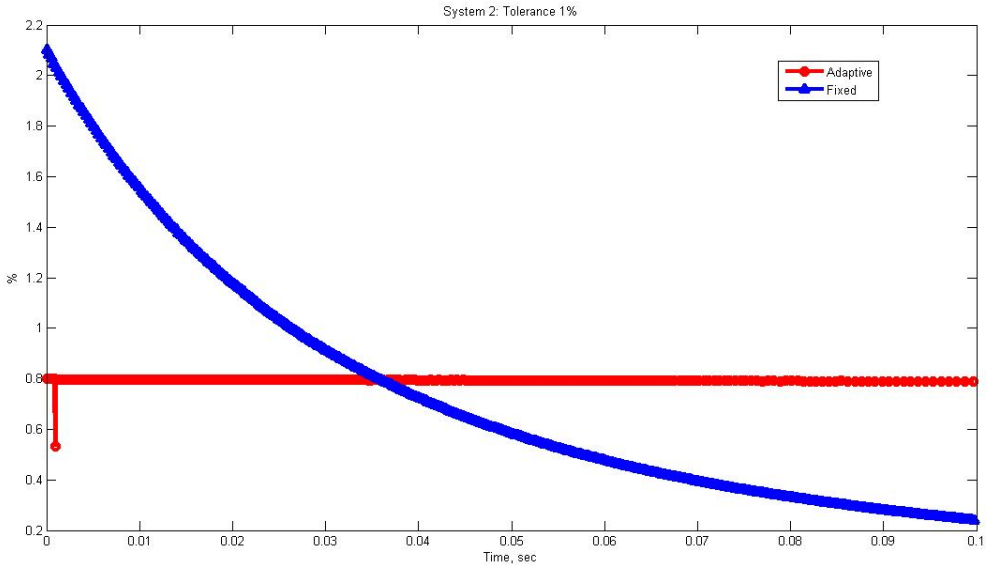

Figure 4.10: System 2, decaying dimerization: plot of the error evolution in time, tolerance = 1%

algorithms wonderfully showcases the dual action of adaptive steps. The adaptive step size not only decreases when more accuracy is required, but also quickly recovers to maintain the error almost equal

to the indicated tolerance.

In Figures 4.11-4.12 we plotted the system dynamics as estimated by the adaptive and the fixed step size algorithms.

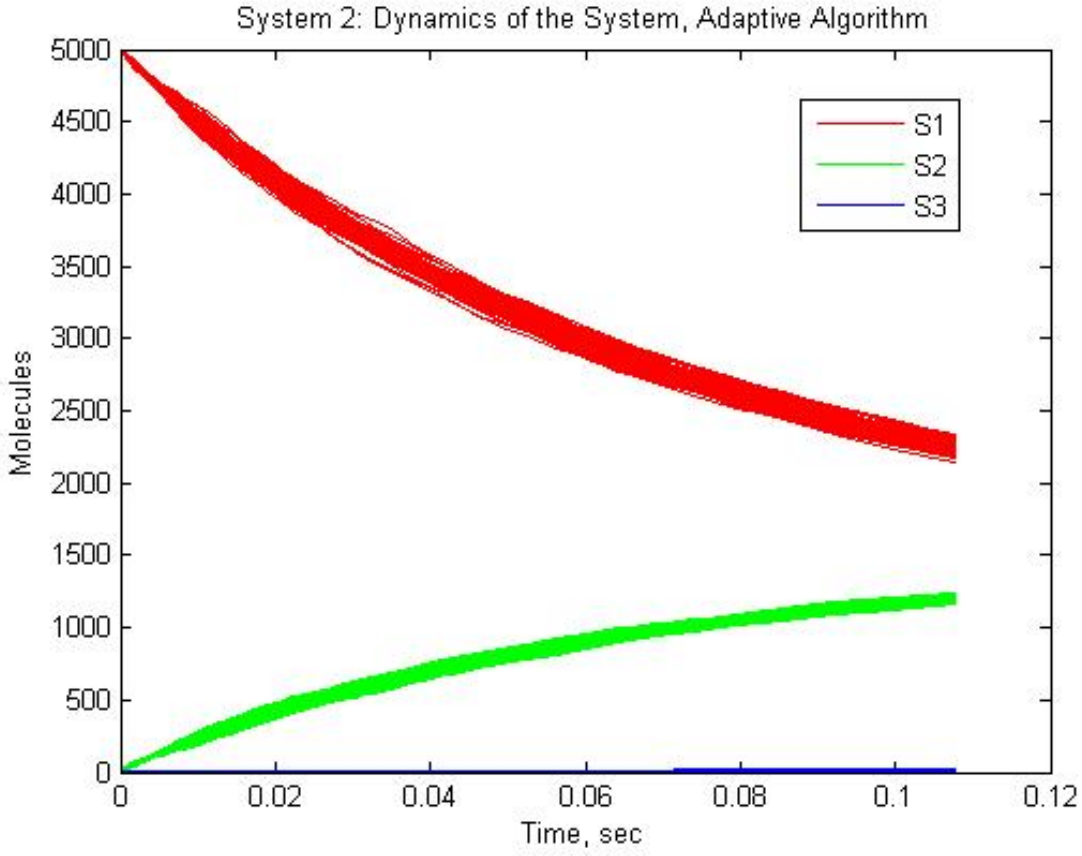

Figure 4.11: System 2, decaying dimerization: plot of the system's dynamics, adaptive step algorithm

From Figures 4.11 and 4.12 we see that, similarly to the Michaelis-Menten model, the adaptive step size algorithm produces a simulation of system's dynamics similar to the one computed by the fixed step algorithm.

4.3 System 3

Finally, the third model we analyze was introduced in [18]. The problem exhibits the most pronounced dynamically stiff behavior. Its reaction rates range from 1e6 molecules/sec to 1e2. We have modified

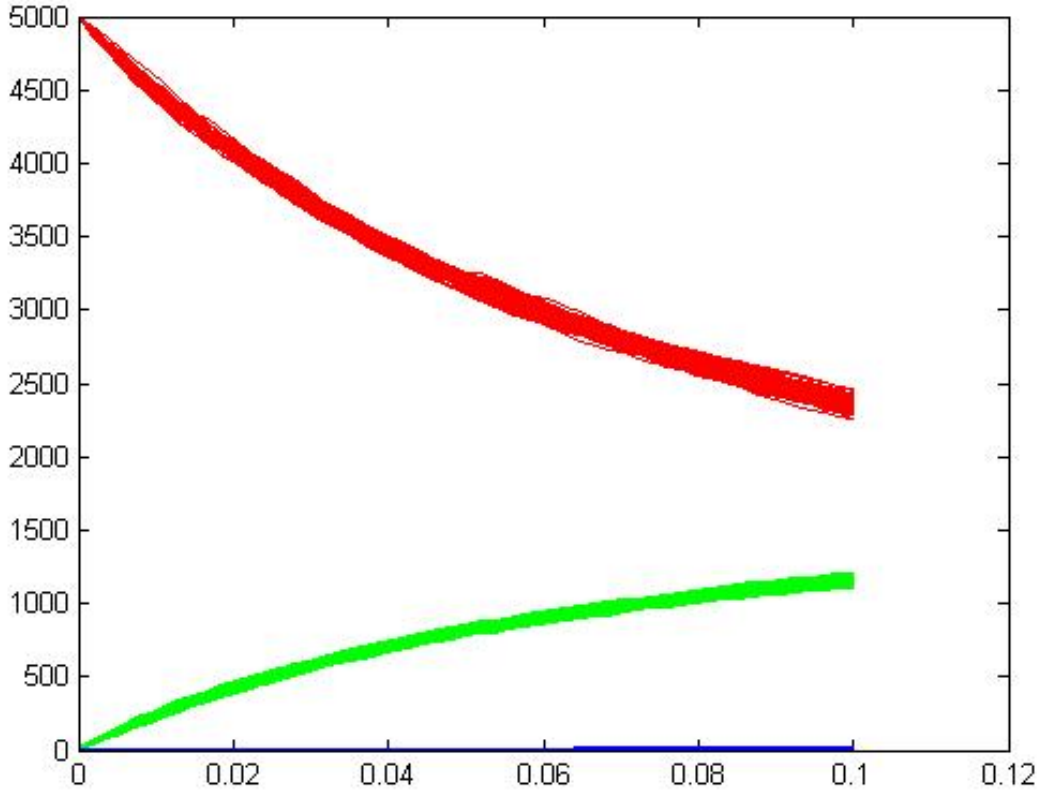

Figure 4.12: System 2, decaying dimerization: plot of the system's dynamics, fixed step algorithm

the coefficients and the initial conditions to make the system less stiff. However, we still observe a considerable difference in the reaction rate magnitudes and thus expect the adaptive time step algorithm to perform better than the fixed step one. The following system of chemical reactions describes the

system:

$$\begin{aligned} S_1 + S_2 &\xrightarrow{c_1} S_3 \\ S_3 &\xrightarrow{c_2} S_1 + S_2 \\ S_1 + S_3 &\xrightarrow{c_3} S_2 \\ S_2 &\xrightarrow{c_4} S_1 + S_3 \\ S_3 + S_2 &\xrightarrow{c_5} S_1 \\ S_1 &\xrightarrow{c_6} S_3 + S_2 \end{aligned}$$

The data we used: the parameters are $C = [1e1, 1e1, 1e-6, 1, 1e-2, 1e4]$, while the initial conditions are $X(t_0) = [100, 99, 10001]$. We ran the simulation on the short time interval $[0, 0.00015]$ sec using 100 trajectories. The stoichiometric matrix associated with this biochemically reacting system is:

$$V = \begin{pmatrix} -1 & 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 \end{pmatrix}$$

The propensities corresponding to the reaction rates:

$$\begin{aligned} a_1(X(t)) &= c_1 X_1(t) X_2(t) \\ a_2(X(t)) &= c_2 X_3(t) \\ a_3(X(t)) &= c_3 X_1(t) X_3(t) \\ a_4(X(t)) &= c_4 X_2(t) \\ a_5(X(t)) &= c_5 X_3(t) X_2(t) \\ a_6(X(t)) &= c_6 X_2(t) \end{aligned}$$

We compared the performance of the adaptive step-size algorithm with that of the regular fixed-step size Langevin Leaping algorithm. We measured the error for the fixed step size algorithm and that for the adaptive algorithm in a similar way.

	Tol, %	No of Steps	No of steps rejected	$\frac{\max error}{tolerance}$
Adaptive	20	73	0	1.000
Fixed	20	73	0	1.778
Adaptive	10	140	0	1.000
Fixed	10	140	0	1.854
Adaptive	5	271	0	1.000
Fixed	5	271	0	1.915
Adaptive	2.5	548	2	1.000
Fixed	2.5	548	0	1.894

Table 4.3: System 3: comparative analysis between the adaptive step and the fixed step algorithms

Table 4.3 summarizes the results for the various runs of the adaptive step algorithm and the constant step one for the System 3. Also, we used the same strategy for comparison of the two algorithms. From this table one can see that given the equal number of steps, the error computed with a constant step mesh is larger than the error obtained on the adaptive mesh. Moreover, as the tolerance decreases, the error to tolerance ratio for the fixed step size grows even more, while the error for the variable step scheme stays the same.

Figures 4.13-4.16 describe the error computed with the two algorithms in time.

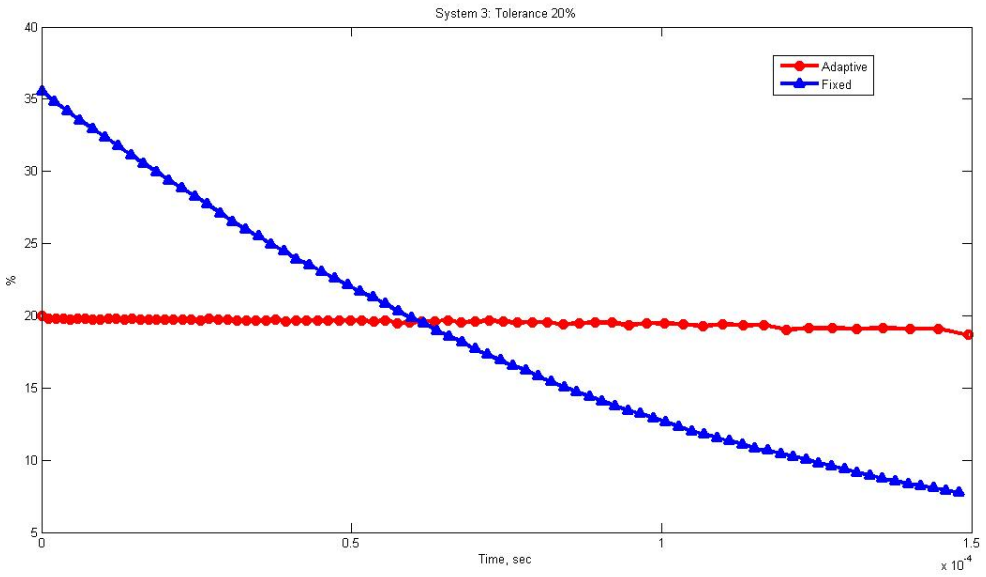

Figure 4.13: System 3: plot of the error evolution in time, tolerance = 20%

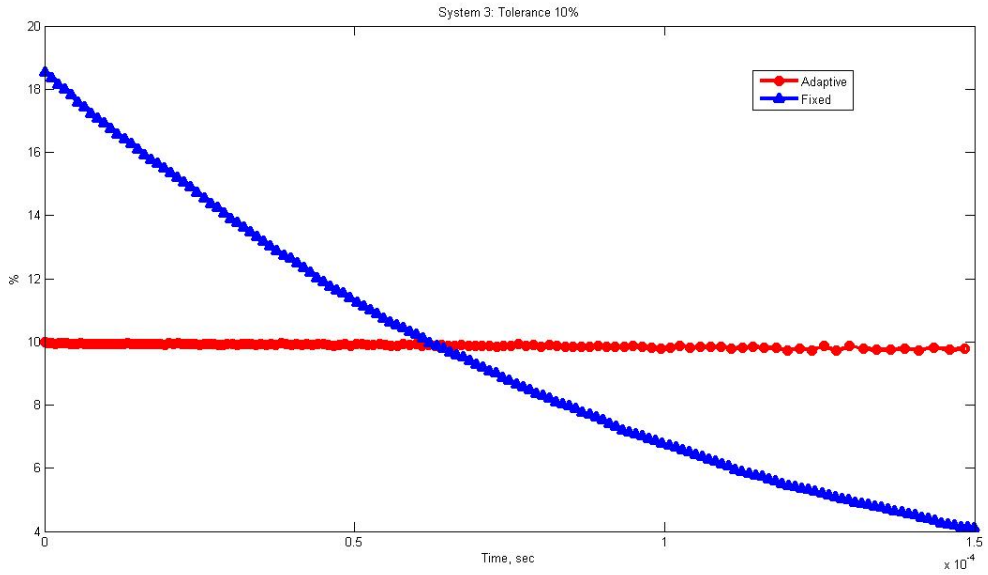

Figure 4.14: System 3: plot of the error evolution in time, tolerance = 10%

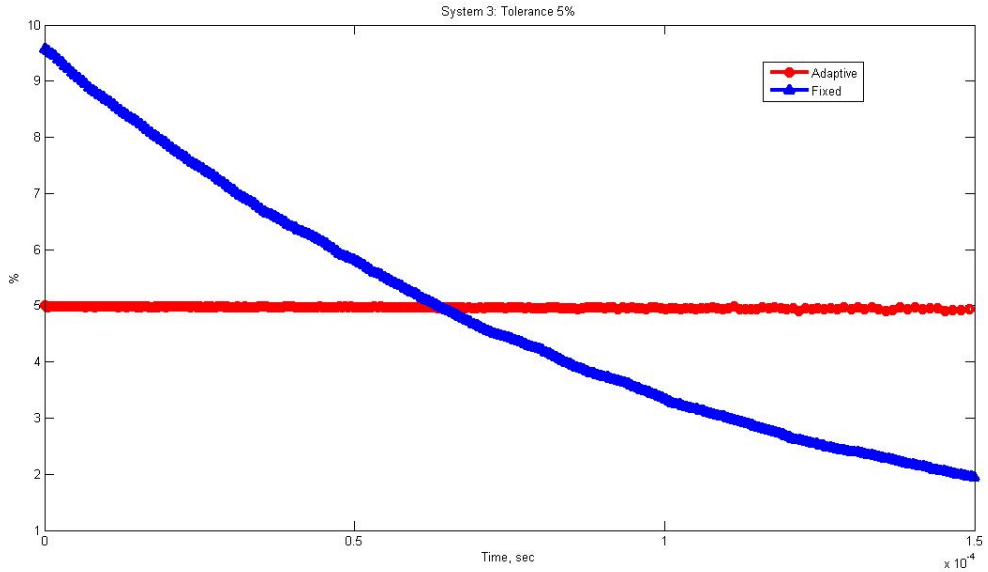

Figure 4.15: System 3: plot of the error evolution in time, tolerance = 5%

This model's dynamics exhibit the same pattern as System 2. From these graphs we see that the maximum error for the fixed step size algorithm occurs during the first 0.5 sec of the simulation. Thus,

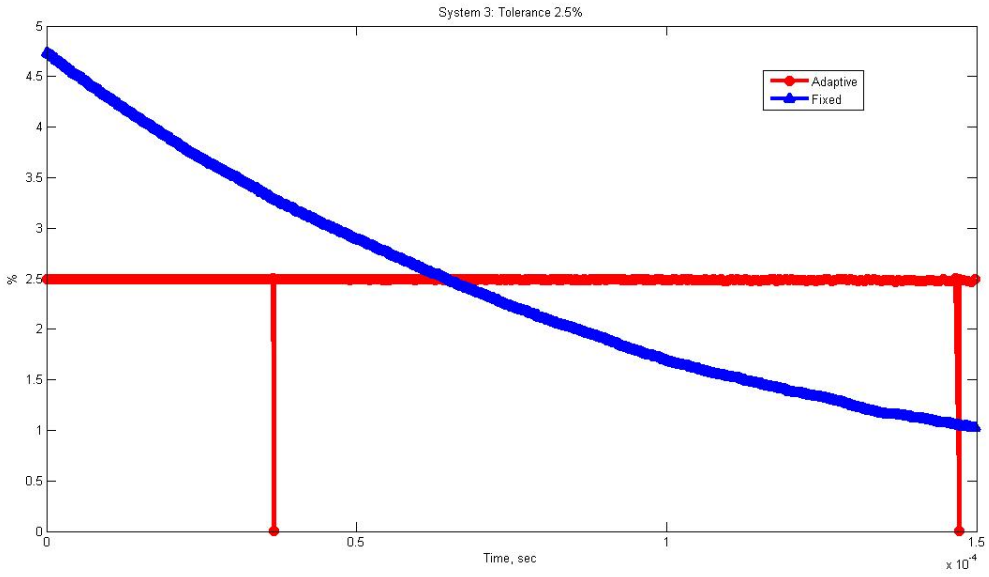

Figure 4.16: System 3: plot of the error evolution in time, tolerance = 2.5%

to have a lower error we should take a smaller step size with the fixed step scheme in the second part of the simulation, after the transient stage.

Figures 4.17-4.18 plot the system dynamics as estimated by the adaptive and the fixed step size algorithms:

From Figures 4.17 and 4.18 we see that, as for the the first and second systems, the adaptive step size algorithm produces a similar behavior as the one produced by the fixed step algorithm.

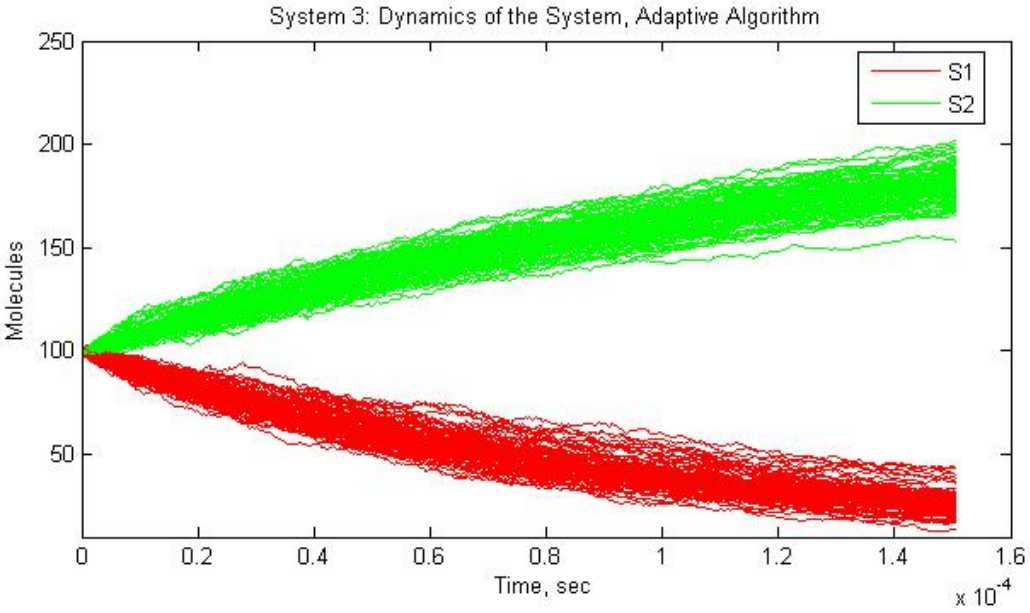

Figure 4.17: System 3: plot of the system's dynamics, adaptive step algorithm

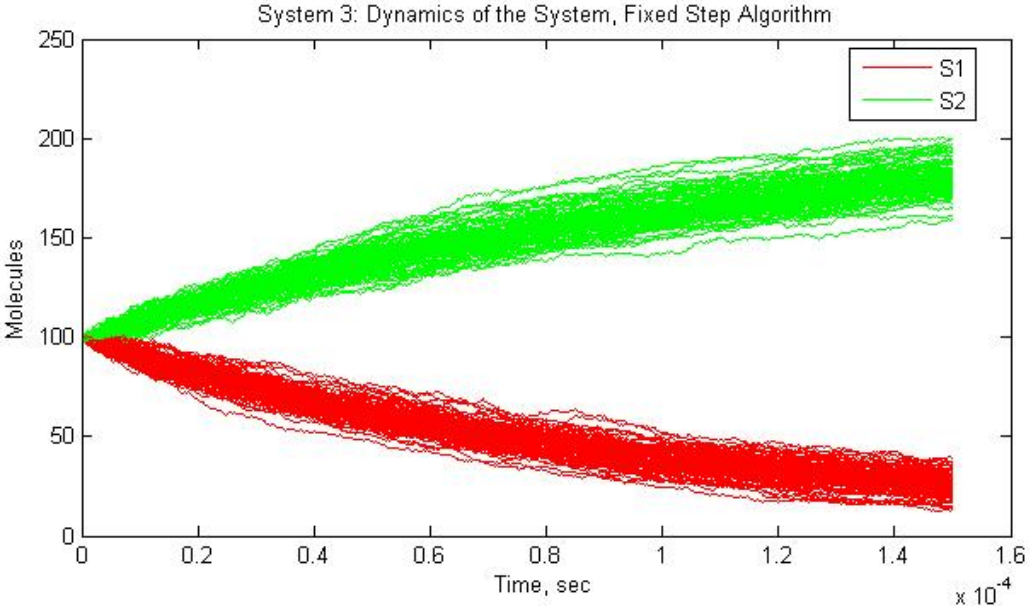

Figure 4.18: System 3: plot of the system's dynamics, fixed step algorithm

Chapter 5

Conclusion

A fundamental problem in the post-genomic biology is to describe and analyze the complex dynamical interactions which take place in the cell. Recent experimental techniques made it possible to study the biological networks in living cells. A successful approach to studying the dynamics associated with these networks should be based on sound mathematical models and powerful new computational tools to simulate them.

There are currently several levels of refinement used for modeling the cellular dynamics. Often chemical kinetic models represent cellular processes as systems of chemical reactions. Traditionally, these processes were modeled as continuous deterministic systems, by ordinary differential equations. However, when some molecular species have relatively low population numbers, then the effects of the random molecular interactions may no longer be neglected. Then stochastic models are required to accurately capture the system's dynamics. The study of stochastic mathematical models of biochemical systems is a critical, but difficult task. As a consequence, the development of new methods and computer tools to deal with the complexity of their simulations is becoming very important.

This thesis focused on the development of efficient numerical methods for approximating the solution of a stochastic continuous mathematical model of biochemical systems. This model is known as the Chemical Langevin Equation. This thesis proposed an adaptive step size algorithm for the numerical solution of the Chemical Langevin Equation. The underlining numerical technique used was the Euler-Maruyama scheme. The adaptive algorithm was compared to the existing fixed step size methods on several problems of interest, and shown to perform much better than those. We have seen a significant

improvement in the running time, while the tolerance criterion was met by the adaptive scheme. The performance of the adaptive algorithm had increasing efficiency, compared to the fixed step algorithm, which grows when the tolerance decreases.

The proposed numerical method will make possible a more efficient and accurate simulation of the proposed models. As a consequence, it will help validate the model and make predictions of its behavior. The approach we propose is based on theoretically rigorous foundations.

Our future work will consider a number of improvements of our algorithm, such as an optimal choice of the initial step of the mesh. We wish to extend the algorithm to estimate the p-mean local error, for an arbitrary p. In addition, we plan to extend the applicability of the adaptive algorithm to problems which are very stiff.

Chapter 6

Appendix A: Code

6.1 Main Programs

6.1.1 Adaptive Step Size Algorithm

Input: Initial state of the system, stoichiometric matrix, reaction rate vector, final time of the integration, first proposed step size, user provided tolerance and total number of trajectories.

Output: Time mesh, the state of the system as a function of time for the number of trajectories indicated by the user, mean error as a function of time, number of rejected steps, number of accepted steps.

Function: Once supplied with all the necessary values for the input, the code initializes the variables, such as the system state variable X and the time vector T. Note that X contains the data for all the trajectories for all mesh points. The counter variable is initialized to the value 1 (not to 0, since Matlab does not support 0-th array indexing). Once the loop begins, it runs until the current value of T(i) is equal to the specified final time. We move the system in time by updating the three variables: X, T and i. Every iteration of the loop starts from generating a set of Wiener increments for the current value of tau and the error estimate eta(i). We generate the Wiener increments only if special flag f_tau is set to true. That means we do not have any previously generated and yet unused Wiener increments associated with the current time step. We set the flag variable f_tau to true. The candidate time step is being generated based on the current error estimate. If the eta(i) is not larger than the user-specified

tolerance, then the variables X, T, and i are update according to the Euler-Maruyama scheme and the execution goes to the first line of the loop. If the current error estimate is larger than the tolerance, it is broken down into smaller steps. First of them, tau1, is equal to the new time step generated previously and, based on the current error estimate and the second step, tau2, equals to difference between the old time step and the new time step. We generate new arrays of Wiener increments associated with these time steps and conditioned on the previously generated Wiener increments for the old time step. The counter of rejected time steps is increased by 1. We move the time variable by tau1, update the state variable X(i+1) according to the Euler-Maruyama scheme, where the time step taken is tau1. Finally, we update the counter i. Having finished with tau1, we proceed to tau2. We calculate the error associated with it, given the current system state. This is done since most likely it is the biggest part of the old time step and very likely to have large error associated with it. We calculate the candidate time step, based on the current error and the step size tau2. If the error is smaller than the tolerance, we move the time variable, T, by tau2, update the system according to the Euler-Maruyama scheme with the time step equal to tau2. Finally, we update the counter variable i, set the time step tau to be equal to the candidate time step and move to the first line of the cycle. In case the error is larger than the tolerance, the time step tau is set to be equal to tau2, and flag f_tau is set to false, which notifies the code that the current step already has Wiener increments generated and not used. We increase by one our counter of rejected steps.

```
function [T,X,eta,rejected,i] = Langevine_keep_traj_v4(S,V,C,Time,tau,tol, total_t)
  %Input check
  %V1 reactant part of stochiometric matrix
  format long;
  n_species = length(S);
  if n_species<=0
     error('Initial State Vector should be non-empty');
  end
  [n1,n_reactions] = size(V);
  if (n1>n_species)
      error('Not all the species used in reactions were given initial state value');
```

```
elseif (n1<n_species)
    error('Initial state vector is too long');
end
n2 = length(C);
if (n2>n_reactions)
    error('Not all of the reactions have been described in change state matrix V');
elseif(n2<n_reactions)
    error('Reaction rate vector is too long');
end
if (tau<0 || Time<0)
    error('Time parameters should be positive');
end
if (tau==0 || Time==0)
   error('System will make no progress');
end
if(tol<=0)
    error('Tolerance should be greater then 0 in order for the system to progress');
end
if (total_t<=0)
   error('Number of trajectories should be a positive number');
end
%Initialization
rejected = 0;
p=2;
gamma = 1;
theta = 1;
%initialization of tau arrays and it's counter of depth c
%for c=1,2,... tau1(c) = tau1(c+1)+tau2(c+1)
%once we are done with depth c+1 we mo to depth c, until c=1
%initialization of system matrix
```

```
i=1;
X = zeros(total_t,2000,n_species);
eta = zeros(1,2000);
x_candidate = zeros(total_t,n_species);
r= zeros(total_t,n_reactions);
r_t= zeros(total_t,n_reactions);
r1=zeros(total_t,n_reactions);
r2=zeros(total_t,n_reactions);
for j=1:total_t
  for k=1:n_species
    X(j,i,k) = S(k);
  end
end

%Symbolic Computation of Propensity Functions
for i=1:n_species
  x(i) = sym(['x' int2str(i)]);
end
for j=1:n_reactions
  a(j) = sym(['a' int2str(j)]);
  a(j) = C(j);
  for i=1:n_species
    if V(i,j)<0
      for k=1:abs(V(i,j))
        a(j) = a(j)*(x(i)-k+1)/k;


      end
    end
  end
   b(j,1) = a(j);
```

```
end
f = V*(b);
f_anal = jacobian(f,x)*f;
Jac = jacobian(a,x);
%i-1 number of leaps system has performed
i=1;
T(i) = 0;

%Main Loop
tau_in = tau;
f_tau = true;
while(T(i)<Time)

  %tau_c = tau;
  %T_c = T(i);
  if (f_tau)
    for j=1:total_t
      for k=1:n_reactions
        r(j,k) = randn*sqrt(tau);
      end
    end
  end
  f_tau=true;
  x_prev = x_previous(X,total_t,i,n_species);
  eta(i) = eta_calc(x,x_prev,total_t,p,tau,f_anal);
  [i T(i) eta(i)]
  %calculation of candidate tau
  tau_new = tau*(theta*tol/eta(i))^(1/gamma);

  if eta(i)<=tol %step was accepted
```

```
  for j=1:total_t
    [x_candidate(j,:)] = x_candidate_c2(X(j,i,:),V,x,r(j,:),tau,a,n_reactions,n_species);
    for l=1:n_species
      X(j,i+1,l) = x_candidate(j,l);
    end
  end
  T(i+1) = T(i)+tau;
  %tau=max(tau_new,0.9*tau_in);
  tau=tau_new;
  i=i+1;

else %step was rejected
  rejected = rejected+1;
  tau1 = tau_new;
  tau2 =tau - tau1;

  for j=1:total_t
    for k=1:n_reactions
      r_t(j,k) = randn;%may need fixing
      r1(j,k) = r(j,k)*tau1/tau+r_t(j,k)*sqrt(tau1*(tau-tau1)/tau);
      r2(j,k) = (tau-tau1)*r(j,k)/tau-r_t(j,k)*sqrt(tau1*(tau-tau1)/tau1);
    end
  end

  eta(i) = eta_calc(x,x_prev,total_t,p,tau1,f_anal);

  for j=1:total_t
    [x_candidate(j,:)] = x_candidate_c2(X(j,i,:),V,x,r1(j,:),tau1,a,n_reactions,n_species);
    for l=1:n_species
      X(j,i+1,l) = x_candidate(j,l);
```

```
  end
end



T(i+1) = T(i)+tau1;
i=i+1;


x_prev = x_previous(X,total_t,i,n_species);
eta(i) = eta_calc(x,x_prev,total_t,p,tau2,f_anal);
tau_new = tau2*(theta*tol/eta(i))^(1/gamma);
if(eta(i)<=tol)%tau2 passed and we have completed this step
  for j=1:total_t
    [x_candidate(j,:)] = x_candidate_c2(X(j,i,:),V,x,r2(j,:),tau2,a,n_reactions,n_species);
    for l=1:n_species
      X(j,i+1,l) = x_candidate(j,l);
    end
  end
  T(i+1) = T(i)+tau2;
  %tau=max(tau_new,0.9*tau_in);
  tau = tau_new;
  i=i+1;


else%tau2 failed and will now serve as tau
  rejected = rejected+1;
  tau=tau2;
  f_tau = false;
  for j=1:total_t
    for k=1:n_reactions
      r(j,k) = r2(j,k);
    end
```

```
        end
      end
    end
  end
```

6.1.2 Fixed Step Size Algorithm

Input: Initial state of the system, stoichiometric matrix, reactions rates vector, final time of the integration, first proposed step size, user provided tolerance and total number of trajectories.

Output: Time mesh, the state of the system as the function of time for the number of trajectories indicated by the user, mean error as a function of time and number of rejected steps.

Function: The program below requires the same input values as the adaptive step algorithm. It proceeds with the execution of the Euler-Maruyama scheme as it was described in Chapter 3. The system's state, the time and the counter variables are updated after each iteration. The execution stops once the current value of T(i) is larger or equal to the final time supplied by the user. For each iteration, we calculate the associated error estimate, however the step size is not being adjusted and remains the same throughout the execution.

```
function [T,X,eta] = Langevine_keep_traj_fix(S,V,C,Time,tau,tol, total_t)
  %Input check
  n_species = length(S);
  if n_species<=0
     error('Initial State Vector should be non-empty');
  end
  [n1,n_reactions] = size(V);
  if (n1>n_species)
      error('Not all the species used in reactions were given initial state value');
  elseif (n1<n_species)
      error('Initial state vector is too long');
  end
  n2 = length(C);
```

```
if (n2>n_reactions)
    error('Not all of the reactions have been described in change state matrix V');
elseif(n2<n_reactions)
    error('Reaction rate vector is too long');
end
if (tau<0 || Time<0)
    error('Time parameters should be positive');
end
if (tau==0 || Time==0)
   error('System will make no progress');
end
if(tol<=0)
    error('Tolerance should be greater then 0 in order for the system to progress');
end
if (total_t<=0)
   error('Number of trajectories should be a positive number');
end
%Initialization
p=2;
gamma = 1;
theta = 0.8;
%initialization of tau arrays and it's counter of depth c
%for c=1,2,... tau1(c) = tau1(c+1)+tau2(c+1)
%once we are done with depth c+1 we mo to depth c, until c=1

%initialization of system matrix
i=1;
num = ceil(Time/tau);
X = zeros(total_t,num,n_species);
eta = zeros(1,num);
```

```
x_candidate = zeros(total_t,n_species);
for j=1:total_t
  for k=1:n_species
    X(j,i,k) = S(k);
  end
end
%set up a flag whether all the future info has been used
r1_f = true;
%Symbolic Computation of Propensity Functions
for i=1:n_species
  x(i) = sym(['x' int2str(i)]);
end
for j=1:n_reactions
  a(j) = sym(['a' int2str(j)]);
  a(j) = C(j);
  for i=1:n_species
    if V(i,j)<0
      for k=1:abs(V(i,j))
        a(j) = a(j)*(x(i)-k+1)/k;

      end
    end
  end
  b(j,1) = a(j);
end
Jac = jacobian(a,x);
f = V*(b);
f_anal = jacobian(f,x)*f;
%i-1 number of leaps system has performed
i=1;
```

```
  T(i) = 0;
  %Main Loop
  while(T(i)<Time)
    x_prev = x_previous(X,total_t,i,n_species);
    eta(i) = eta_calc(x,x_prev,total_t,p,tau,f_anal);
    %calculation  of x_candidate for each trajectory
    for j=1:total_t
      for k=1:n_reactions
        %for each trajectory for each reaction we generating
        r1(j,k) = randn*sqrt(tau);
      end
      %calculation of propensity function sets for each trajectory
      [x_candidate(j,:)] = x_candidate_c2(X(j,i,:),V,x,r1(j,:),tau,a,n_reactions,n_species);
    end
    %calculation of eta(i)
    %calculation of sum_p (over all trajectories)
    for j=1:total_t
      for l=1:n_species
        X(j,i+1,l) = x_candidate(j,l);
      end
    end
    T(i+1) = T(i)+tau;
    i=i+1;
  end
end
```

6.2 Sub-Modules

6.2.1 Error Estimation

Input: Vector of variables x, current sizes of each species population for each trajectory, total number of trajectories, p, time step, expression for function which we will use to evaluate the error estimate.

Output: error estimate associated with the proposed time step.

Function: Estimates the error associated with the proposed time step tau, given the current state of the system for all trajectories.

```
function [eta] = eta_calc(x,x1,total_t,p,tau,f_anal)


  theta_2=0.6;
  n_species = length(x);
  f_anal_c = zeros(n_species,total_t);%jacobian(f,x)*f container
  sum_2p = 0;
  for j=1:total_t
     sum_2p = sum_2p+ sum(subs(f_anal,x,x1(j,1,:)).^p);%calculation of jacobian(f,x)*f for each trajecto
  end


  norm_p =(sum_2p/total_t)^(1/p);%%% mean norm_p across all trajectories
  eta = tau*abs(theta_2-0.5)*norm_p(1,1);%eta calculation based on p-norm and current tau
end
```

6.2.2 Produce candidate update to system's state

Input: Current state of the system on all trajectories, X, stoichiometric matrix V, set of symbolic variables (corresponding to species) x1, set of random variables one per each reaction on each trajectory, r1, perspective time step tau, number of reactions and number of species.

Output: Vector of the system step after the time leap tau.

Function: Provide the state of the system at time t, X(t), calculates the system's state, X(t+tau), at the time t+tau as per the Langevin Leaping algorithm.

```
function [x_candidate] = x_candidate_c2(X,V,x,r1,tau,a,n_reactions,n_species)
  %calculation of propensity function sets for each trajectory
  a0c = subs(a,x,X);
  %calculation of x_candidate
  for l=1:n_species
    x_candidate(l) =  X(l);
    for k=1:n_reactions
      x_candidate(l) =  x_candidate(l) + a0c(k)*tau.*V(l,k)
       +sqrt(a0c(k))*r1(k).*V(l,k);
    end
  end
end
```

6.2.3 View State of the System for all trajectories at any passed point in time

Input: Record of the system dynamics so far, X, total number of trajectories, point in time i, number of the molecular species in the system.

Output: The state of each trajectory for all species at time T(i).

```
function [x_prev] = x_previous(X,total_t,i,n_species)
x_prev = zeros(total_t,n_species);
x_prev = X(:,i,:);
end
```

Chapter 7

Appendix B: Probability

7.1 Poisson and Binomial Distributions

We state that a Binomial Distributed random variable with a large number of trials N and a small probability $\frac{\lambda}{N}$ of invidual success can be approximated by a Poisson distributed random variable with mean λ.

To show this, we note that:

$$\lim_{n\to\infty}(1-\frac{\lambda}{n})^n = e^{-\lambda} \tag{7.1}$$

If X_n is a binomially distributed variable with the probability of individual success equal to λ/n and the number of trials equal to n, then:

$$\lim_{n\to\infty} P(X_n \leq k) = \lim_{n\to\infty} \binom{n}{k}(\lambda/n)^k(1-\lambda/n)^{n-k} = \lim_{n\to\infty} \frac{n!}{k!(n-k)!}(\lambda/n)^k(1-\lambda/n)^n(1-\lambda/n)^{-k} \tag{7.2}$$

Note that,

$$\lim_{n\to\infty}(1-\lambda/n)^{-k} = 1 \tag{7.3}$$

Substituting (7.3) into (7.2), we obtain

$$\lim_{n\to\infty} P(X_n \leq k) = \lim_{n\to\infty} \frac{n!}{n^k(n-k)!}\frac{\lambda^k}{k!}e^{-\lambda} \tag{7.4}$$

Since,

$$\lim_{n\to\infty} \frac{n!}{n^k(n-k)!} = 1 \ , \tag{7.5}$$

then

$$\lim_{n\to\infty} P(X_n \leq k) = \frac{\lambda^k}{k!} e^{-\lambda} \tag{7.6}$$

Equation (7.6) states that indeed a Binomial random variable with parameters $p = \frac{\lambda}{N}$ and N, can be approximated by a Poisson variable with mean equal to λ.

7.2 Poisson and Normal Distributions

We state that a Poisson distributed random variable with a large mean λ can be approximated by a normally distributed variable with the mean and the variance equal to λ. We will give the intuition behind this approximation below.

If we take a Poisson random variable with a large mean λ, we can represent it as a sum of N identical random Poisson variables with mean equal to λ/N. This is due to the fact that if the random variable Z is a sum of two independent Poisson distributed random variables with means μ_1 and μ_2, then Z is a Poisson distributed random variable with mean equal to $\mu_1 + \mu_2$. However, the Central Limit Theorem states that the sum of a relatively large number of independent variables with finite means and variances will be approximately normally distributed. Therefore, a Poisson distributed random variable with a relatively large mean can be approximated by a Normal random variable with the same mean.

References

[1] A. Arkin, J. Ross, and H. McAdams. Stochastic kinetic analysis of development pathway bifurcation in phage λ-infected escherichia coli cells. *Genetics*, 149:1633–1648, 2008.

[2] Y. Cao, D. Gillespie, and L. Petzold. Avoiding negative populations in explicit poisson tau-leaping. *Journal of Chemical Physics*, 123(5), 2005.

[3] Y. Cao, D. Gillespie, and L. Petzold. The slow-scale stochastic simulation algorithm. *Journal of Chemical Physics*, 122, 2005.

[4] Y. Cao, D. Gillespie, and L. Petzold. Efficient step size selection for the tau-leaping simulation method. *Journal of Chemical Physics*, 126, 2006.

[5] Y. Cao, D. Gillespie, and L. Petzold. Adaptive explicit-implicit tau-leaping method with automatic tau selection. *Journal of Chemical Physics*, 126, 2007.

[6] M. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.

[7] D. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

[8] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25), 1977.

[9] D. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4), 2001.

[10] D. Gillespie and L. Petzold. Improved leap-size selection for accelerated stochastic simulation. *Journal of Chemical Physics*, 2003.

[11] D. Higham. Modeling and simulating chemical reactions. *SIAM Review*, 50(2):347–368, 2008.

[12] D. Ingber and I. Tensegrity. Cell structure and hierarchical systems biology. *Journal of Cell Science*, 116(7):1157–1173, 2003.

[13] H. D. Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[14] S. Kar, W. Baumann, M. Paul, and J. Tyson. Exploring the roles of noise in the eukaryotic cell. *PNAS*, 106(16):6471–6476, 2009.

[15] P. Kloeden and E. Platen. *Numerical Solution of STochastic Differential Equation*. Springer, Berlin, 1992.

[16] M. Rathinam, L. Petzold, Y. Cao, and D. Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *Journal of Chemical Physics*, 119(24), 2003.

[17] W. Romisch and R. Winkler. Stepsize control for mean square numerical methods for stochastic differential equations with small noise. *SIAM Journal of Scientific Computing*, 28(2), 2006.

[18] V. Sotiropoulos and Y. Kaznessis. An adaptive time step scheme for a system of stochastic differential equations with multiple multiplicative noise: Chemical langevin equation, a proof of concept. *Journal of Chemical Physics*, 128, 2008.

[19] T. Tian and K. Burrage. Binomial leap methods for simulating stochastic chemical kinetics. *Journal of Chemical Physics*, 121(21), 2004.

[20] T. Turner, S. Schnell, and K. Burrage. Stochastic approaches for modeling in vivo reactions. *Computational Biology and Chemistry*, 28:165–178, 2004.

[21] J. Vilar, H. Kueh, N. Barkai, and S. Leibler. Mechanisms of noise-resistance in genetic oscillators. *PNAS*, 99(9):5988–5991, 2002.