# Numerical Analysis of Stochastic Biochemical Reaction Networks

A dissertation submitted towards the degree
Doctor of Engineering (Dr.-Ing.)
of the Faculty of Mathematics and Computer Science
of Saarland University

by

Linar Mikeev

Saarbrücken
2017

| | |
|---|---|
| Day of Colloquium | 31 January 2018 |
| Dean of the Faculty | Univ.-Prof. Dr. Frank-Olaf Schreyer |
| Chair of the Committee | Prof. Dr. Holger Hermanns |
| Reporters | |
| First reviewer | Prof. Dr. Verena Wolf |
| Second reviewer | Prof. Dr. Luca Bortolussi |
| Third reviewer | Dr.-Ing. Jan Hasenauer |
| Academic Assistant | Dr. Daniel Stan |

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

# Acknowledgements

I would like to express my deep gratitude to Prof. Dr. Verena Wolf, my research supervisor, for her enthusiastic encouragement and patient guidance throughout the course of this research work. Her constructive suggestions and willingness to dedicate her time have been very much appreciated. I am profoundly grateful to Dr. Werner Sandmann for introducing me to the topic of rare events and for his valuable input on our study on approximative numerical solution methods. Without his aid and continuous support, this project would have been impossible. I would also like to express my very great appreciation to Dr. Jan Hasenauer from Helmholtz Zentrum München whom I met when he visited Saarland University and whose work on the method of conditional moments served as inspiration for my research. My grateful thanks are also extended to Prof. Dr. Tuğrul Dayar whom I was lucky to get to know while he was visiting our department during his sabbatical leave from Bilkent University; he guided me at the very beginning of my research with his valuable constructive suggestions. Special thanks should be given to Prof. Dr. Luca Bortolussi from University of Trieste for his useful critiques of this research work.

Additionally, I wish to thank my colleagues, Alexander Andreychenko and David Spieler, for their assistance with the project on the parameter estimation. I would also like to extend my thanks to the Prof. Dr. Hermanns' group for fruitful discussions and knowledge sharing: Christian Eisentraut, Luis María Ferrer Fioriti, Ernst Moritz Hahn, Arnd Hartmanns, Vahid Hashemi, Hassan Hatefi, Martin R. Neuhäußer, Lei Song and Andrea Turrini. My thanks also go to everyone whom I had the opportunity to meet at conferences and workshops, including researchers from the ROCKS project. And finally, I wish to thank my parents for their support throughout my research work.

# Abstract

Numerical solution of the chemical master equation for stochastic reaction networks typically suffers from the state space explosion problem due to the curse of dimensionality and from stiffness due to multiple time scales. The dimension of the state space equals the number of molecular species involved in the reaction network and the size of the system of differential equations equals the number of states in the corresponding continuous-time Markov chain, which is usually enormously huge and often even infinite. Thus, efficient numerical solution approaches must be able to handle huge, possibly infinite and stiff systems of differential equations efficiently.

In this thesis, we present efficient techniques for the numerical analysis of the biochemical reaction networks. We present an approximate numerical integration approach that combines a dynamical state space truncation procedure with efficient numerical integration schemes for systems of ordinary differential equations including adaptive step size selection based on local error estimates. We combine our dynamical state space truncation with the method of conditional moments, and present the implementation details and numerical results. We also incorporate ideas from importance sampling simulations into a non-simulative numerical method that approximates transient rare event probabilities based on a dynamical truncation of the state space. Finally, we present a maximum likelihood method for the estimation of the model parameters given noisy time series measurements of molecular counts. All approaches presented in this thesis are implemented as part of the tool STAR, which allows to model and simulate the biochemical reaction networks. The efficiency and accuracy is demonstrated by numerical examples.

# Zusammenfassung

Numerische Lösungen der chemischen Master-Gleichung für stochastische Reaktionsnetzwerke leiden typischerweise an dem Zustandsraumexplosionsproblem aufgrund der hohen Dimensionalität und der Steifigkeit durch mehrfache Zeitskalen. Die Dimension des Zustandsraumes entspricht der Anzahl der molekularen Spezies von dem Reaktionsnetzwerk und die Größe des Systems von Differentialgleichungen entspricht der Anzahl der Zustände in der entsprechenden kontinuierlichen Markov-Kette, die in der Regel enorm gross und oft sogar unendlich gross ist. Daher müssen numerische Methoden in der Lage sein, riesige, eventuell unendlich grosse und steife Systeme von Differentialgleichungen effizient lösen zu können.

In dieser Arbeit beschreiben wir effiziente Methoden für die numerische Analyse biochemischer Reaktionsnetzwerke. Wir betrachten einen inexakten numerischen Integrationsansatz, bei dem eine dynamische Zustandsraumbeschneidung und ein Verfahren mit einem effizienten numerischen Integrationsschema für Systeme von gewöhnlichen Differentialgleichungen benutzt werden. Wir kombinieren unsere dynamische Zustandsraumbeschneidungsmethode mit der Methode der bedingten Momente und beschreiben die Implementierungdetails und numerischen Ergebnisse. Wir benutzen auch Ideen des importance sampling für eine nicht-simulative numerische Methode, die basierend auf der Zustandsraumbeschneidung die Wahrscheinlichkeiten von seltenen Ereignissen berechnen kann. Schließlich beschreiben wir eine Maximum-Likelihood-Methode für die Schätzung der Modellparameter bei verrauschten Zeitreihenmessungen von molekularen Anzahlen. Alle in dieser Arbeit beschriebenen Ansätze sind in dem Software-Tool STAR implementiert, das erlaubt, biochemische Reaktionsnetzwerke zu modellieren und zu simulieren. Die Effizienz und die Genauigkeit werden durch numerische Beispiele gezeigt.

# Contents

# List of Figures

# List of Tables

# Introduction

The chemical master equation (CME) describes the random dynamics of many stochastic reaction networks in physics, biology, and chemistry, amongst other sciences. More specifically, many biochemical reaction networks can be appropriately modeled by multivariate continuous-time Markov chains (CTMCs), also referred to as Markov jump processes, where at any time the system state is represented by a vector of the numbers of each molecular species present in the reaction network and the CME is a system of differential equations whose solution, given an initial probability distribution, provides the transient (time-dependent) state probabilities.

Exact analytical solutions of the CME are only available for very small reaction networks or special cases such as, e.g., monomolecular reactions or reversible bimolecular reactions [JH07, Lau00]. Therefore, in general computational approaches are required. However, because reaction rates typically differ by several orders of magnitude, the system dynamics possess multiple time scales and the corresponding equations are stiff. Furthermore, the size of the state space typically increases exponentially with the model dimensionality, that is, with the number of molecular species in the reaction network. This effect is often referred to as the curse of dimensionality or state space explosion. Stochastic simulation of the reaction network and the numerical solution of the CME are two common complementary approaches to analyze stochastic reaction networks governed by the CME, both of which must be properly designed to cope with huge, potentially infinite multidimensional state spaces, and stiffness.

Stochastic simulation does not solve the CME directly but imitates the reaction network dynamics by generating trajectories (sample paths) of the underlying CTMC, where a stochastically exact imitation is in

principle straightforward [BKL75, Gil76, Gil77]. Several mathematically equivalent implementations have been developed [CLP04, GB00, LP06, MPC+06, San08]. Stochastic simulation does not suffer from state space explosion, because the state space needs not be explicitly enumerated, but in particular stochastic simulation of stiff systems becomes exceedingly slow and inefficient, because simulating all successive reactions in order to generate trajectories of the underlying CTMC advances only in extremely small time steps. In order to tackle this problem approximate stochastic simulation techniques for accelerated trajectory generation have been developed, in particular various tau-leaping methods [And08, CGP06, CGP07, Gil01, RES07, RPCG03, San09c, TB04, XC08] where, instead of simulating every reaction, at any time $t$ a time step size $\tau$ is determined by which the simulation is advanced.

A general problem of stochastic simulation, however, remains also with approximate techniques: stochastic simulation constitutes an algorithmic statistical estimation procedure that tends to be computationally expensive and only provides estimates whose reliability and statistical accuracy in terms of relative errors or confidence interval half widths depend on the variance of the corresponding simulation estimator. Estimating the whole probability distribution by stochastic simulation is enormously time consuming. Therefore, often only statistical estimates of the expected numbers of molecules are considered, which requires less effort, but in any case, when estimating expectations, probabilities, or any other relevant system property, many stochastically independent and identically distributed trajectories must be generated in order to achieve a reasonable statistical accuracy [San09b]. Hence, stochastic simulation is inherently costly. In many application domains it is sometimes even referred to as a method of last resort.

Several hybrid approximation approaches combine stochastic simulation with deterministic numerical computations. One way to do this is by distinguishing between low molecular counts, where the evolution is described by the CME and handled by stochastic simulation, and high molecular counts, where an approximation of the CME by the continuous-state Fokker-Planck equation is viable [HL07, SSDN15]. Another way is to consider time scale separations where parts of the system (states or reactions) are classified as either slow or fast and the different parts are handled by different stochastic simulation approaches and numerical solution techniques [BTB04, WLVE07, HC06, HR02, MBBS08, MBS08,

RA03, SK05, VB04], or even without resorting to stochastic simulation for any part [BSW06, SW08]. As a major drawback, however, for time scale separation methods it is in general hard to define what is slow or fast. In fact, many systems possess multiple time scales rather than only two and a clear separation of time scales is impossible.

Some numerical approximation approaches tackle the state space explosion problem by restricting the analysis of the model to certain subsets of states where the truncated part of the state space has only a sufficiently small probability. For instance, finite state projection (FSP) algorithms [BHM$^+$06, MK06, MK07] consider finite parts of the state space that can be reached either during the whole time period of interest or during multiple time intervals into which the time period of interest is split. Then the computation of transient probabilities is conducted based on the representation of the transient probability distribution as the product of the initial probability distribution times a matrix exponential involving the generator matrix of the underlying CTMC restricted to the finite projection. However, computing matrix exponentials is well known to be an intricate issue in itself [MVL78, MVL03] and with FSP algorithms it must be repeated multiple times, corresponding to repeated expansions of the finite state projection. Alternative state space truncation methods are based on adaptive wavelet methods [Jah10, JU10], or on a conversion to discrete time where it is dynamically decided which states to consider at a certain time step in a uniformized discrete-time Markov chain [MWDH10].

While examples show that these methods can handle the state space explosion in some cases, for many systems they are still not feasible, because even the considered finite part of the state space is too large, or because the specific system dynamics hamper the computational methods involved. In particular, even if the state space is relatively small or can be reduced to a manageable size, the stiffness problem must be handled satisfactorily and it is not clear whether these methods are suitable for stiff systems.

In Chapter 1, we introduce the notation for stochastic reaction networks, the underlying CTMC and in particular the CME that describes the transient probability distribution.

In Chapter 2, we present a novel numerical approximation method that tackles both the state space explosion problem and the stiffness problem

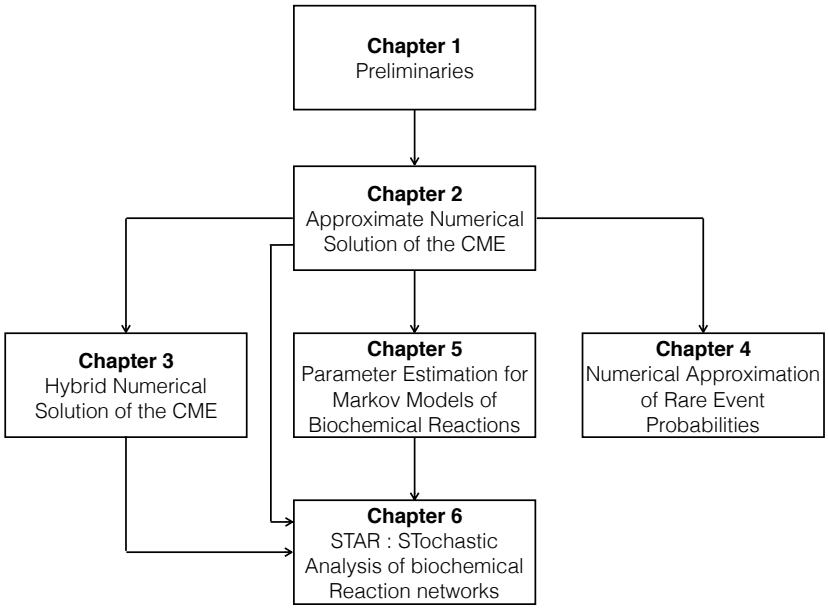



Figure 1 – Chapter dependencies.

where we consider efficient approximate numerical integration based on the fact that the CME can be cast in the form of a system of ordinary differential equations (ODEs) and the computation of transient probabilities, given an initial probability distribution, is an initial value problem (IVP). We combine a dynamical state space truncation procedure, efficient numerical integration schemes, and an adaptive step size selection based on local error estimates. The dynamical state space truncation keeps the number of considered states manageable while incurring only a small approximation error. It is much more flexible and can reduce the state space much more than the aforementioned methods. The use of efficient ODE solvers with adaptive step size control ensures that the method is fast and numerically stable by taking as large as possible steps without degrading the method's convergence order.

In Chapter 3, we present in detail an efficient algorithm for the hybrid solution of the CME based on the method of conditional moments. We describe the combined integration of moment and state probability equations and allow the truncation of insignificant states. We illustrate the accuracy and efficiency of the algorithm by a number of examples for which we compare to a pure method of moments solution and to a direct numerical solution of the CME.

In Chapter 4, we address the computation of rare event probabilities in stochastic models of biochemically reacting systems and Markovian queueing networks with huge or possibly even infinite state spaces. For this purpose, we incorporate ideas from importance sampling simulations into a non-simulative numerical method that approximates transient probabilities based on a dynamical truncation of the state space. A change of measure technique is applied in order to accomplish a guided state space exploration. Numerical results for different example networks demonstrate the efficiency and accuracy of our method.

In Chapter 5, we propose a numerical technique for parameter inference in Markov models of biological processes. Based on time-series data of a process we estimate the kinetic rate constants, initial populations and parameters representing measurement errors by maximizing the likelihood of the data. The computation of the likelihood relies on a dynamic abstraction of the discrete state space of the Markov model which successfully mitigates the problem of state space largeness. We compare two variants of our method to state-of-the-art, recently published methods and demonstrate their usefulness and efficiency on several case studies from systems biology. Moreover, we provide more complex case studies and run extensive numerical experiments to assess the identifiability of certain parameters. In these experiments we assume that not all molecular populations can be observed and estimate parameters for different observation scenarios, i.e., we assume different numbers of observed cells and different observation interval lengths.

Finally, in Chapter 6, we present the tool STAR (STochastic Analysis of Reaction networks), which combines a variety of efficient methods for stochastic modeling and simulation of complex biochemical reaction networks. Models can be imported from SBML or can be specified using a simple human readable language, whereas the simulation methods can be accessed via the lightweight and easy-to-use scripting language Lua. Along with the standard sampling methods, the tool provides efficient numerical methods for the computation of probability distributions and statistical moments of the model by solving a system of differential equations. Moreover, the tool efficiently computes derivatives of the probability distribution w.r.t. model parameters and uses them for sensitivity analysis and gradient-based optimization methods.

Figure 1 depicts the chapter dependencies.

# 1 Preliminaries

In this chapter, we introduce stochastic models of biochemical reaction networks which are used throughout the thesis. We also introduce the underlying CTMC that describes the transient probability distribution. Finally, we introduce the Gillespie stochastic simulation algorithm (SSA) which is commonly used for stochastic simulation of biochemical reaction networks. We assume that the reader is familiar with the basics of the probability theory and the theory of differential equations.

## 1.1 Chemical reaction networks

Consider a well-stirred mixture of $N \in \mathbb{N}$ molecular species $S_1, \ldots, S_N$ in a thermally equilibrated system of fixed volume, interacting through $R \in \mathbb{N}$ different types of chemical reactions, also referred to as chemical reaction channels, $\mathcal{R}_1, \ldots, \mathcal{R}_R$. At any time $t \geq 0$ a discrete random variable $X_i(t)$ describes the number of molecules of species $S_i$ and the system state is given by the random vector $\mathbf{X}(t) = (X_1(t), \ldots, X_N(t))$.

The system changes its state due to one of the possible reactions, where each reaction channel $\mathcal{R}_j, j = 1, \ldots, R$, is defined by a stoichiometric equation

$$\mathcal{R}_j : \; v_{j,1}^- S_1 + \ldots + v_{j,N}^- S_N \xrightarrow{c_j} v_{j,1}^+ S_1 + \ldots + v_{j,N}^+ S_N, \tag{1.1}$$

with associated stoichiometric coefficients $v_{j,1}^-, \ldots, v_{j,N}^-$, $v_{j,1}^+, \ldots, v_{j,N}^+$ and stochastic reaction rate constant $c_j$. Mathematically, the stoichiometry is described by the state change vector $\mathbf{v}_j = \mathbf{v}_j^+ - \mathbf{v}_j^- = (v_1, \ldots, v_N)$, where $v_{j,i}$ is the change of molecules of species $S_i$ due to $\mathcal{R}_j$. That is, if a

reaction of type $\mathcal{R}_j$ occurs when the system is in state $\mathbf{x}$, then the next state is $\mathbf{x} + \mathbf{v}_j$, or, equivalently, state $\mathbf{x}$ is reached, if a reaction of type $\mathcal{R}_j$ occurs when the system is in state $\mathbf{x} - \mathbf{v}_j$.

For each reaction channel $\mathcal{R}_j$ the reaction rate is given by a state-dependent propensity function $\alpha_j$, where $\alpha_j(\mathbf{x})dt$ is the conditional probability that a reaction of type $\mathcal{R}_j$ occurs in the time interval $[t, t + dt)$, given that the system is in state $\mathbf{x}$ at time $t$. That is

$$\alpha_j(\mathbf{x})dt = Pr(\mathcal{R}_j \text{ occurs in } [t, t + dt) \mid \mathbf{X}(t) = \mathbf{x}) \,. \tag{1.2}$$

The propensity function is given by $c_j$ times the number of possible combinations of the required reactants and thus computes as

$$\alpha_j(\mathbf{x}) = c_j \prod_{i=1}^{N} \binom{x_i}{v_{j,i}^-}, \tag{1.3}$$

where $x_i$ is the number of molecules of species $S_i$ present in state $\mathbf{x}$, and $v_{j,i}^-$ is the stoichiometric coefficient of $S_i$ according to (1.1). Because at any time the system's future evolution only depends on the current state, $(\mathbf{X}(t))_{t\geq 0}$ is a time-homogeneous continuous-time Markov chain (CTMC), also referred to as a Markov jump process, with $N$-dimensional state space $\mathcal{X} \subseteq \mathbb{N}_0^N$. Since the state space is countable it is always possible to map it to $\mathbb{N}$, which yields a numbering of the states.

Table 1.1 – Chemical reactions of the simple enzyme reaction network.

$$\begin{aligned} \mathrm{E} + \mathrm{S} &\underset{c_2}{\overset{c_1}{\rightleftharpoons}} \mathrm{C} \\ \mathrm{C} &\xrightarrow{c_3} \mathrm{E} + \mathrm{P} \end{aligned}$$

As an example, we consider a simple enzyme reaction network with three reactions given in Table 1.1 that involve four different species, namely enzymes (E), substrates (S), complex molecules (C), and proteins (P). The reactions are complex formation (E + S $\rightarrow$ C), dissociation of the complex (C $\rightarrow$ E + S), and protein production (C $\rightarrow$ E + P). The corresponding rate functions are $\alpha_1(\mathbf{x}) = c_1 \cdot x_1 \cdot x_2$, $\alpha_2(\mathbf{x}) = c_2 \cdot x_3$, and $\alpha_3(\mathbf{x}) = c_3 \cdot x_3$ where $\mathbf{x} = (x_1, x_2, x_3, x_4)$. Here, $x_1$ denotes the number of enzymes, $x_2$ the number of substrates, $x_3$ the number of complex molecules, and $x_4$ the number of product molecules. The change vectors are given as $\mathbf{v}_1^- = (1, 1, 0, 0)$, $\mathbf{v}_1^+ = (0, 0, 1, 0)$, $\mathbf{v}_2^- = (0, 0, 1, 0)$, $\mathbf{v}_2^+ = (1, 1, 0, 0)$, $\mathbf{v}_3^- = (0, 0, 1, 0)$, and $\mathbf{v}_3^+ = (1, 0, 0, 1)$.

## 1.2 Transition classes

*Transition class* formalism can be used to describe a Markovian event system which, in particular, can describe a biochemical reaction network. It defines the state space of the system and specifies all relevant events that may trigger state transitions. Also it defines under which conditions a certain event may occur, how it affects the system state and at which rate it occurs. Diverse formal specifications of Markovian event systems can be found in the literature. Here, we adopt the transition class formalism of [San04b]. Without loss of generality we assume that the state space is $\mathcal{X} \subseteq \mathbb{N}^N$. All events that trigger state transitions are classified according to their effects which yields transition classes. Formally, a transition class is a triplet $\mathcal{C}_j = (\mathcal{U}_j, u_j, \alpha_j)$ where $\mathcal{U}_j \subseteq \mathbb{N}^N$ is the source state space containing all states in which the event or the corresponding state transition, respectively, is possible, $u_j : \mathcal{U}_j \to \mathbb{N}^N$ is the update function giving the new state $u_j(\mathbf{x}) \in \mathbb{N}^N$ according to the state transition when the event occurs in state $\mathbf{x} \in \mathcal{U}_j$, and $\alpha_j : \mathcal{U}_j \to \mathbb{R}$ is the transition rate function giving the rate $\alpha_j(\mathbf{x}) \in \mathbb{R}$ at which the event or transition occurs in state $\mathbf{x} \in \mathcal{U}_j$. A particularly appealing feature is that very different systems can be cast in the same formalism. Any Markovian event system, in particular any network of biochemical reactions, can be uniquely described by a set of such transition classes together with an initial distribution. The numbering of this set is arbitrary but often closely reflects the actual meaning of the transitions. Thus, a biochemical reaction network containing $R$ reactions can be described using $R$ transition classes, where each reaction channel $\mathcal{R}_j$ corresponds to a transition class $\mathcal{C}_j = (\mathcal{U}_j, u_j, \alpha_j)$ with $\mathcal{U}_j = \{(x_1, \ldots, x_N) \in \mathbb{N}^N : x_i - v_{j,i}^- \geq 0\}$ and $u_j(\mathbf{x}) = (x_1 + v_{j,1}, \ldots, x_N + v_{N,1})$.

## 1.3 Chemical master equation

The conditional transient (time-dependent) probability that the system is in state $\mathbf{x} \in \mathcal{X}$ at time $t$, given that the system starts in an initial state $\mathbf{x}_0 \in \mathcal{X}$ at time $t_0$, is denoted by

$$p_{\mathbf{x}}(t) := p(\mathbf{x}, t) := p(\mathbf{x}, t|\mathbf{x}_0, t_0) = Pr(\mathbf{X}(t) = \mathbf{x} \mid \mathbf{X}(t_0) = \mathbf{x}_0) \quad (1.4)$$

and the transient probability distribution at time $t$ is the collection of all transient state probabilities at that time, represented by the row vector

$\mathbf{p}(t)$. The system dynamics in terms of the state probabilities' time derivatives are described by the chemical master equation (CME)

$$\frac{d}{dt}p(\mathbf{x},t) = \mathcal{M}(\mathbf{p}(t))(\mathbf{x}), \tag{1.5}$$

where the operator $\mathcal{M}$ is defined for any real-valued function $g : \mathbb{N}^N \to \mathbb{R}$ such that $\mathcal{M}(g)$ is the function that maps a state $\mathbf{x}$ to the value

$$\mathcal{M}(g)(\mathbf{x}) = \sum_{\substack{j=1 \\ \mathbf{x}\geq v_j^+}}^{R} \left(\alpha_j(\mathbf{x}-\mathbf{v}_j)g(\mathbf{x}-\mathbf{v}_j,t) - \alpha_j(\mathbf{x})g(\mathbf{x},t)\right).$$

Equation (1.5) is also well known as the system of Kolmogorov forward differential equations for Markov processes [VK92] and can be written in the vector-matrix form as

$$\frac{d}{dt}\mathbf{p}(t) = \mathbf{p}(t)Q, \tag{1.6}$$

where $Q$ is the infinitesimal generator matrix of $\mathbf{X}$ with $Q(\mathbf{x}_1,\mathbf{x}_2) = \alpha_j(\mathbf{x}_1)$ if $\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{v}_j$ and $\mathbf{x}_1 - \mathbf{v}_j^- \geq \mathbf{0}$. Note that, in order to simplify our presentation, we assume here that all vectors $\mathbf{v}_j$ are distinct. All remaining entries of $Q$ are zero except for the diagonal entries which are equal to the negative row sum. These stochastic reaction kinetics are physically well justified since they are evidently in accordance with the theory of thermodynamics [Gil92, VK92]. In the thermodynamic limit the stochastic description converges to classical deterministic mass action kinetics [Kur72].

Note that (1.5) is the most common way to write the CME, namely as a partial differential equation (PDE), where $t$ as well as $x_1,\ldots,x_N$ are variables. However, for any fixed state $\mathbf{x} = (x_1,\ldots,x_N)$ the only free parameter is the time parameter $t$ such that (1.5) with fixed $\mathbf{x}$ is an ordinary differential equation (ODE) with variable $t$. In particular, when solving for the transient state probabilities numerical ODE solvers can be applied. We shall therefore use the notation $p_{\mathbf{x}}(t)$ in the following.

**Algorithm 1** Stochastic simulation algorithm.

1: $t \leftarrow 0$, $\mathbf{x} \leftarrow \mathbf{x}_0$
2: **while** $t < T$ **do**
3: $\quad \alpha_0 \leftarrow \sum_{j=1}^{R} \alpha_j(\mathbf{x})$
4: $\quad$ **if** $\alpha_0 = 0$ **then break**
5: $\quad$ sample $r_1, r_2 \sim U(0,1)$
6: $\quad \tau \leftarrow \frac{1}{\alpha_0(\mathbf{x})} ln\left(\frac{1}{r_1}\right)$
7: $\quad j \leftarrow$ the smallest integer satyisfying $\sum_{j=1}^{R} \alpha_j(\mathbf{x}) > r_2 \alpha_0(\mathbf{x})$
8: $\quad \mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}_j$, $t \leftarrow t + \tau$
9: **end while**

## 1.4 Stochatic simulation algorithm

The Gillespie stochastic simulation algorithm (SSA) is commonly used for the simulation of biochemical reaction networks [Gil76, Gil77]. It generates a trajectory which represents an exact sample from the probability mass function that is the solution of the master equation (1.5). The pseudocode is given in Algorithm 1. Initially, the state of the system is set to $\mathbf{x}_0$. In each iteration, we compute the exit rate which is the sum of rates of all possible reactions in the current state $\mathbf{x}$. Then we determine the next reaction to occur with the probability of $j$-th reaction to occur of $\frac{\alpha_j(\mathbf{x})}{\alpha_0(\mathbf{x})}$. We also sample the time interval after which the next reaction occurs according to the exponential distribution with the parameter $\alpha_0(\mathbf{x})$. Finally, we update the simulation time and the state of the system according to the chosen reaction. We continue iterating until the final simulation time $T$ is reached or no more reactions are possible in the current state (line 4 in the pseudocode).

Denote by $t_1 < t_2 < \ldots$ the successive time instants at which reactions occur and by $\mathcal{R}_{j_i}, j_i \in \{1, R\}$ the reaction that applies at time $t_i$. Let $\tau_i := t_{i+1} - t_i$ be the time between the $i$-th and the $(i+1)$-th reaction. Hence, state $\mathbf{x}(t_i)$ is reached due to the $i$-th reaction according to $\mathcal{R}_{j_i}$ at time $t_i$ and remains unchanged for a sojourn time of $\tau_i$ after which the $(i+1)$-th transition according to $\mathcal{R}_{j_{i+1}}$ occurs at time $t_{i+1}$ and changes the state to $\mathbf{x}(t_{i+1})$. Thus, the time evolution of the system is completely described by the sequence of states and corresponding sojourn times. In compact form, $(\mathbf{x}(t_0), \tau_0), (\mathbf{x}(t_1), \tau_1), (\mathbf{x}(t_2), \tau_2), \ldots$ describes a trajectory where $t_0 := 0$ and $\tau_0 = t_1$ is the sojourn time in the initial state $\mathbf{x}(0)$.

For a trajectory up to the $K$-th transition, considering the Markovian

property which in turn implies exponentially distributed sojourn times, the path density is given by

$$dP((\mathbf{x}(t_0),\tau_0),\ldots,(\mathbf{x}(t_K),\tau_K)) = \\ p^{(t_0)}(\mathbf{x}(0)) \cdot \prod_{i=1}^{K} \alpha_{j_i}\left(\mathbf{x}(t_{i-1})\right) \exp\left(-\alpha_0(\mathbf{x}(t_{i-1}))\tau_{i-1}\right) \quad (1.7)$$

where $\alpha_0(\mathbf{x}(t_{i-1})) := \alpha_1(\mathbf{x}(t_{i-1})) + \cdots + \alpha_R(\mathbf{x}(t_{i-1}))$ is the parameter (reciprocal mean) of the exponential sojourn time in state $\mathbf{x}(t_{i-1}), i = 1,\ldots,K$. Note that for a given time horizon the number $K$ of transitions is not known in advance and not deterministic. Formally, it is a random stopping time, which is in accordance with $dP$ being a density of a probability measure $P$ defined on the path space of the Markov process.

# 2 Approximate Numerical Solution of the CME

Numerical integration methods for solving the CME, given $\mathbf{p}^{(0)} := \mathbf{p}(t_0)$, discretize the integration interval $[0, T]$ and successively compute approximations $\mathbf{p}^{(1)} \approx \mathbf{p}(t_1), \mathbf{p}^{(2)} \approx p(t_2), \ldots, p^{(\eta)} \approx p(t_\eta)$, where $0 = t_0 < t_1 < t_2 \cdots < t_\eta = T$ are the mesh points and $h_i = t_{i+1} - t_i$ is the step size at the $i$-th step, $i = 0, \ldots, \eta - 1$. With single-step methods each approximation $\mathbf{p}^{(i+1)} \approx \mathbf{p}(t_{i+1})$ is computed in terms of the previous approximation $\mathbf{p}^{(i)}$ only, that is, without using approximations $\mathbf{p}^{(j)}, j < i$. For advanced methods the step sizes $h_i$ (and thus $\eta$, the number of steps) are not determined in advance, but variable step sizes are determined in the course of the iteration.

The system of ODEs described by the CME (1.5) is typically large or even infinite, because there is one ODE for each state in the underlying CTMC, that is, the size of the system of ODEs equals the size of the CTMC's state space. Thus, its solution with standard numerical integration methods becomes computationally infeasible. However, one can exploit that at any time only a tractable number of states have "significant" probability, that is, only relatively few states have a probability that is greater than a small threshold.

## 2.1 Dynamical state space truncation

The main idea of our dynamical state space truncation for numerical integration methods is to integrate only those differential equations in the CME (1.5) that correspond to significant states. All other state probabilities are (temporarily) set to zero. This reduces the computational

effort significantly since in each iteration step only a comparatively small subset of states is considered. Based on the fixed probability threshold $\delta > 0$, we dynamically decide which states to drop or add, respectively. Due to the regular structure of the CTMC the approximation error of the algorithm remains small since probability mass is usually concentrated at certain parts of the state space. The farther away a state is from a "significant set" the smaller is its probability. Thus, in most cases the total error of the approximation remains small. Since in each iteration step probability mass may be "lost" the approximation error at step $i$ is the sum of all probability mass lost (provided that the numerical integration could be performed without any errors), that is,

$$1 - \sum_{\mathbf{x} \in \mathcal{X}} p_{\mathbf{x}}^{(i)}. \tag{2.1}$$

It is important to note that other than static state space truncation approaches our dynamical approach allows that in the course of the computation states can "come and go", that is, states join the significant set if and only if their current probability is above the threshold $\delta$ and states in the significant set are dropped immediately when their current probability falls below $\delta$. Furthermore, states that have previously been dropped may come back, that is they are re-considered as significant as soon as they receive a probability that exceeds the threshold $\delta$. This is substantially different from state space exploration techniques where only the most probable states are generated but states are never dropped as time progresses like for instance in [dSeSO92] with regard to approximating stationary distributions. Our dynamical state space truncation approach is also much more flexible than finite state projection (FSP) algorithms [BHM+06, MK06, MK07] which work over pre-defined time intervals with the same subset of states, where in particular for stiff systems many reactions can occur during any time interval, so that in order to safely meet reasonable accuracy requirements the resulting subset of states is often still extremely large. In contrast to that we update our set of significant states in each adaptively chosen time step, without much overhead. Furthermore, by numerically integrating the ODEs we avoid the intricate computation of matrix exponentials required in FSP algorithms and by using an efficient data structure we do not even need to generate any matrices.

In order to avoid the explicit construction of a matrix and in order to

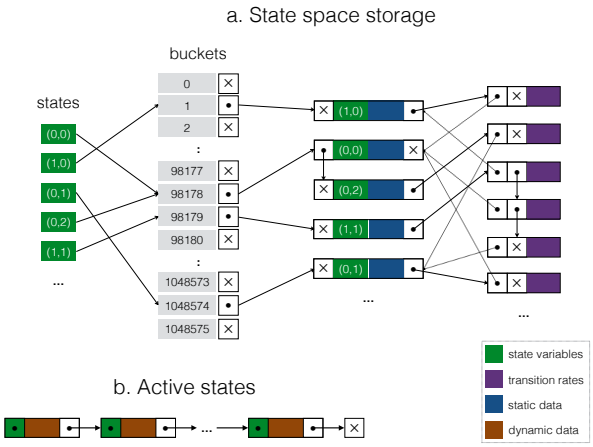



Figure 2.1 – State space storage: a) hash table storing the information about the successor states, b) list of active states with significant probability mass ($> \delta$).

work with a dynamic set *Sig* of significant states that changes in each step, we use a data structure depicted in Figure 2.1. We store the set *Sig* of significant states, i.e. active states, as a doubly-linked list where each element contains a pointer to the element in the hash table which represents the state (Figure 2.1a) and "dynamic" data which is updated over the simulation time and is not required after the state is removed from the list of active states. This data includes state probabilities and other intermediate variables used during the numerical integration. Note that each state is mapped into a bucket in the hash table using a hash function, and the collisions are handled using a separate chaining with linked lists. Each entry of the hash table representing a state has the description of the state (i.e. the state variables), so that it can be verified that an entry corresponds to a state when doing a lookup. We also store for each state in the hash table a pointer to an element in the list of active states, if it is currently in the set *Sig* of significant states. Also, we store a pointer to the first successor state or a special value if the successor states haven't been added yet. Finally, we store some state related "static" data (e.g. its index) which does not change over the integration time. Each element of the list of successors contains a pointer to the element in the hash table representing the successor state, and also contains data related to the transition to the successor state (e.g. transition rate).

In Figure 2.1a, the hash table contains 5 states: $(0, 0)$, $(1, 0)$, $(0, 1)$, $(0, 2)$ and $(1, 1)$. States $(0, 0)$ and $(0, 2)$ are mapped into the same bucket due to the hash collision, and therefore, stored in the same linked list. The

**Algorithm 2** A numerical adaptive step size integration scheme.

```
1: Sig ← {x : p_x(0) > δ}
2: t ← 0, i ← 0
3: compute h_0
4: while t < T do
5:     compute p^(i+1)
6:     compute ε̂_i, h_{i+1}
7:     if step successful then
8:         update Sig
9:         t ← t + h_i, i ← i+1
10:    end if
11: end while
```

transitions (on the right) correspond to transitons: from state $(1,0)$ to state $(0,0)$, from state $(0,2)$ to state $(0,1)$, from state $(1,1)$ to state $(1,0)$, from state $(1,1)$ to state $(0,0)$, from state $(1,1)$ to state $(0,1)$, and from state $(0,1)$ to state $(0,0)$. Note that state $(1,1)$ has multiple successor states which are stored in the same linked list.

The numerical integration algorithm only operates on the list of active states. Initially, a set of states is added to the list of active states, meaning that for each state we also add a corresponding element to the hash table. Note that the allocation of elements in the list of active states and in the hash table is efficiently implemented using fixed size memory pools. When doing numerical integration, the successors are firstly looked up in the hash table, and if they are not set, the successor states are evaluated using the transition function. Each successor state is then looked up in the hash table before adding it to the hash table. Note that we only store reachable states in the list of successor states which makes it memory efficient in case of a high number of transition classes. Removing a state only requires several pointer assignments. The corresponding state in the hash table is not removed, only the pointer to the element in the list of active states is set to null.

The workflow of the numerical integration scheme is given in pseudocode in Algorithm 2. We start at time $t = 0$ and initialize the set *Sig* as the set of all states that have initially a probability greater than $\delta$. We compute the initial time step $h_0$. In each iteration step we compute the approximation $p^{(i+1)}$ using an explicit or implicit Runge-Kutta method (see Sections 2.2, 2.3). We check whether the iteration step was successful

computing the local error estimate $\hat{\epsilon}_i$ and ensuring that error tolerance conditions are met. If so, then for each state we update the field $\mathbf{x}.p$ with $\mathbf{x}.p_2$, and remove the state from *Sig* if its probability becomes less than $\delta$. Based on the local error estimate, we choose a time step for the next iteration (or the repetition of the iteration in case it was not successful). This and the computation of the initial time step is detailed in Section 2.4.

## 2.2 Explicit methods

We consider the whole family of Runge-Kutta methods, which proceed each time step of given step size in multiple *stages*. More precisely, a general $s$-stage Runge-Kutta method proceeds according to the iteration scheme

$$p^{(i+1)} = p^{(i)} + h_i \sum_{\ell=1}^{s} b_\ell k^{(\ell)}, \tag{2.2}$$

$$k^{(\ell)} := \mathcal{M}\left(p^{(i)} + h_i \sum_{j=1}^{s} a_{\ell j} k^{(j)}\right), \tag{2.3}$$

which is uniquely defined by weights $b_1 \ldots, b_s > 0$ with $b_1 + \cdots + b_s = 1$ and coefficients $0 \leq a_{\ell j} \leq 1$, $\ell = 1, \ldots, s,\ j = 1, \ldots, s$. Thus, $k^{(\ell)}$ is an approximation to $p(t_i + h_i c_\ell)$, where $c_\ell = a_{\ell 1} + \cdots + a_{\ell s}$, and $k^{(\ell)}(x)$ is the component of the vector $k^{(\ell)}$ that corresponds to state $x$. Hence, $k^{(\ell)}(x)$ is the probability of $x$ at stage $\ell$. If $a_{\ell j} = 0$ for all $j \geq \ell$, then the sum in Equation (2.3) effectively runs only from 1 to $\ell - 1$, which means that for each $\ell = 1, \ldots, s$ the computation of $k^{(\ell)}$ includes only previous stage terms $k^{(j)}, j < \ell$. Therefore, $k^{(1)}, \ldots, k^{(s)}$ can be computed sequentially, that is, $a_{\ell j} = 0$ for all $j \geq \ell$ yields explicit integration schemes. If there is at least one $j \geq \ell$ with $a_{\ell j} > 0$, then the integration scheme is implicit, which implies that the solution of at least one linear system of equations is required per iteration step.

A single iteration step for general explicit $s$-stage Runge-Kutta schemes is given in pseudocode in Algorithm 3. We denote the approximated probabilities $p_{\mathbf{x}}^{(i)}$ and $p_{\mathbf{x}}^{(i+1)}$, as $\mathbf{x}.p$ and $\mathbf{x}.p_2$ respectively, We also use fields $\mathbf{x}.k_1, \ldots, \mathbf{x}.k_s$ as the stage terms $k^{(1)}(x), \ldots, k^{(s)}(\mathbf{x})$ and initialize

**Algorithm 3** A single iteration step of a general explicit $s$-stage Runge-Kutta scheme, defined by $s, b_1, \ldots, b_s$, and $a_{\ell j}$ for $\ell = 1, \ldots, s,\ j < \ell$, with dynamical state space truncation, which approximates the solution of the CME.

1: **for** $\ell \leftarrow 1$ to $s$ **do**
2: **for all** $\mathbf{x} \in \mathit{Sig}$ **do**
3: $\hat{p} \leftarrow x.p + h \cdot \sum_{j=1}^{\ell-1} a_{\ell j} \cdot \mathbf{x}.k_j$
4: **for all** $j \in \{1, \ldots, R\} : x + \mathbf{v}_j \geq \mathbf{0}$ **and** $\left(\{\mathbf{x} + \mathbf{v}_j\} \in \mathit{Sig}$ **or** $h \cdot \alpha_j(\mathbf{x}) \cdot \hat{p} > \hat{\delta}\right)$ **do**
5: **if** $\{\mathbf{x} + \mathbf{v}_j\} \notin \mathit{Sig}$ **then** $\mathit{Sig} \leftarrow \mathit{Sig} \cup \{\mathbf{x} + \mathbf{v}_j\}$ **end if**
6: $\mathbf{x}.k_\ell \leftarrow \mathbf{x}.k_\ell - h \cdot \alpha_j(\mathbf{x}) \cdot \hat{p}$
7: $(\mathbf{x} + \mathbf{v}_j).k_\ell \leftarrow (\mathbf{x} + \mathbf{v}_j).k_\ell + h \cdot \alpha_j(\mathbf{x}) \cdot \hat{p}$
8: **end for**
9: **end for**
10: **end for**
11: **for all** $\mathbf{x} \in \mathit{Sig}$ **do**
12: $\mathbf{x}.p_2 \leftarrow \mathbf{x}.p + h \cdot \sum_{j=1}^{s} b_j \cdot \mathbf{x}.k_j$
13: $\mathbf{x}.k_1 \leftarrow 0, \ldots, \mathbf{x}.k_s \leftarrow 0$
14: **end for**

them with zero. We compute the approximation of $p^{(i+1)}$ based on Equation (2.2) by traversing the set *Sig* $s + 1$ times. In the first $s$ rounds (lines 1-10) we compute $\mathbf{x}.k_1, \ldots, \mathbf{x}.k_s$ according to Equation (2.3) and in the final round (lines 11-14) we accumulate the summands and zero $x.k_1, \ldots, x.k_s$. While processing state $\mathbf{x}$ in round $\ell \leq s$, for each reaction channel $j$, we transfer probability mass from state $\mathbf{x}$ to its successor $\mathbf{x} + \mathbf{v}_j$, by subtracting a term from $\mathbf{x}.k_i$ and adding the same term to $(\mathbf{x} + \mathbf{v}_j).k_i$ (lines 6-7). We do so after checking (line 4) whether $x + v_m$ is already in *Sig*, and if not, whether it is worthwhile to add $\mathbf{x} + \mathbf{v}_j$ to *Sig*, that is, we guarantee that $\mathbf{x} + \mathbf{v}_j$ will receive enough probability mass and that $\mathbf{x} + \mathbf{v}_j$ will not be removed in the same iteration. Thus, we add $\mathbf{x} + \mathbf{v}_j$ to *Sig* (line 5) only if the inflow $h \cdot \alpha_j(x) \cdot (\mathbf{x}.p + h \cdot \sum_{j=1}^{\ell-1} a_{\ell j} \cdot \mathbf{x}.k_j)$ to $\mathbf{x} + \mathbf{v}_j$ is greater or equal than a certain threshold $\hat{\delta} > 0$. Obviously, $\mathbf{x} + \mathbf{v}_j$ may receive more probability mass from other states and the total inflow may be greater than $\hat{\delta}$. Thus, if a state is not a member of *Sig* and if for each incoming transition the inflow probability is less than $\hat{\delta}$, then this state will not be added to *Sig* even if the total inflow is greater or equal than $\hat{\delta}$. This small modification yields a significant speed-up since otherwise all states that are reachable within at most $s$ transitions will always be added to *Sig*, but many of the newly added states will be removed in the

same iteration.

## 2.3 Implicit methods

The advantage of implicit methods is that they can usually take larger (and thus fewer) steps, which comes at the price of an increased computational effort per step, but paying that price can lead to large speed-up of the overall integration over the time interval $[0, T]$. It is common sense that in general the efficient solution of stiff ordinary differential equations requires implicit integration schemes [HW96]. In the present paper, with regard to implicit numerical integration we restrict ourselves to the implicit Euler method, hence the special case of a one-stage Runge-Kutta method with $a_{11} = 1$ and $b_1 = 1$, which yields

$$p^{(i+1)} = p^{(i)} + h_i \mathcal{M}\left(p^{(i+1)}\right) \tag{2.4}$$

and requires to solve the linear system

$$p^{(i+1)} - h_i \mathcal{M}\left(p^{(i+1)}\right) = p^{(i)} \tag{2.5}$$

for $p^{(i+1)}$ in each step.

Of course, when considering a standard approach to the numerical integration of the CME, where no state space truncation is considered, then this linear system is huge, possibly infinite, and its solution is often impossible. In conjunction with our dynamical state truncation procedure, the linear system is reduced similarly to the reduction of the state space and the number of differential equations to be integrated per step, respectively. However, there a subtleties that must be properly taken into account.

Firstly, since we do not need to maintain huge matrices but we use the previously described dynamical data structure the solution of the linear system must be accordingly implemented with this data structure. Secondly, some states that are not significant at time $t$ may receive a significant probability at time $t + h_i$ and must be included in the linear system. Thus, a dynamical implementation of the solution of the linear system is required. Therefore, iterative solution techniques for linear systems that are usually simply defined by a fixed matrix must be properly adapted to the dynamical data structure and the dynamical state space truncation.

**Algorithm 4** A single iteration step of an implicit Euler scheme using the Jacobi method with dynamical state space truncation, which approximates the solution of the CME.

1: **while** convergence not reached **do**
2: **for all** $\mathbf{x} \in Sig$ **do**
3: **for all** $j \in \{1, \ldots, R\} : \mathbf{x}+\mathbf{v}_j \geq \mathbf{0}$ **and** $\left(\{\mathbf{x}+\mathbf{v}_j\} \in Sig$ **or** $h \cdot \alpha_j(\mathbf{x}) \cdot \mathbf{x}.p_1 > \hat{\delta}\right)$ **do**
4: **if** $\{\mathbf{x}+\mathbf{v}_j\} \notin Sig$ **then** $Sig \leftarrow Sig \cup \{\mathbf{x}+\mathbf{v}_j\}$ **end if**
5: $(\mathbf{x}+\mathbf{v}_j).k \leftarrow (\mathbf{x}+\mathbf{v}_j).k + \alpha_j(\mathbf{x}) \cdot \mathbf{x}.p_1$
6: **end for**
7: **end for**
8: **for all** $x \in Sig$ **do**
9: $\mathbf{x}.p_2 \leftarrow (\mathbf{x}.p + h \cdot \mathbf{x}.k)/(1 + h \cdot \alpha_0(\mathbf{x}))$
10: check convergence for state $x$
11: $\mathbf{x}.p_1 \leftarrow \mathbf{x}.p_2$
12: $\mathbf{x}.k \leftarrow 0$
13: **end for**
14: **end while**

In fact, when using implicit numerical integration schemes in conjunction with the dynamical state space truncation procedure, in principle the solution of a linear system in each integration step is a challenging potential bottleneck. It is therefore a key point and a key contribution to implement it efficiently.

We illustrate the solution of the linear system (2.5) using the Jacobi method, which yields the following iterative scheme

$$p_{\mathbf{x}}^{(i+1,j+1)} = \frac{p_{\mathbf{x}}^{(i)} + h_i \cdot \sum_{j=1}^{R} \alpha_j(\mathbf{x}-\mathbf{v}_j) \cdot p_{\mathbf{x}-\mathbf{v}_j}^{(i+1,j)}}{1 + h_i \cdot \alpha_0(\mathbf{x})}, \tag{2.6}$$

where $\alpha_0(\mathbf{x}) = \sum_{j=1}^{R} \alpha_j(\mathbf{x})$. The pseudocode is given in Algorithm 4. In the $(i+1)$-th iteration of the adaptive numerical integration scheme (Algorithm 2), we store the "old" approximation of the state probability $p_{\mathbf{x}}^{(i+1,j)}$ in the field $\mathbf{x}.p_1$ and the "new" approximation $p_{\mathbf{x}}^{(i+1,j+1)}$ in the field $\mathbf{x}.p_2$. We initialize $\mathbf{x}.p_1$ with the state probability from the $i$-th iteration $p_{\mathbf{x}}^{(i)}$. In lines 2-7 for each state we compute the sum $\sum_{j=1}^{R} \alpha_j(\mathbf{x}-\mathbf{v}_j) \cdot p_{\mathbf{x}-\mathbf{v}_j}^{(i+1,j)}$ and store it in a field $\mathbf{x}.k$. While processing state $\mathbf{x}$, for each reaction channel $j$, we transfer probability mass from state $\mathbf{x}$ to its successor $\mathbf{x}+\mathbf{v}_j$. Similarly to Algorithm 3, we only add a new state to $Sig$ if it receives enough probability mass. In lines 9-14 for each state we compute the

"new" approximation of $p_{\mathbf{x}}^{(i+1)}$ according to (2.6) and check whether the convergence criterion

$$|p_{\mathbf{x}}^{(i+1,j+1)} - p_{\mathbf{x}}^{(i+1,j)}| \leq \max(rtol \cdot \max(p_{\mathbf{x}}^{(i+1,j+1)}, p_{\mathbf{x}}^{(i+1,j)}), atol), \quad (2.7)$$

is fulfilled for some relative and absolute tolerances $rtol > 0$ and $atol > 0$. Algorithm 4 terminates if (2.7) holds for all states $\mathbf{x} \in Sig$. After the convergence of the Jacobi method, the field $\mathbf{x}.p_2$ contains the approximation $p_{\mathbf{x}}^{(i+1)}$.

For our numerical experiments we use the Gauss-Seidel method, which is known to converge faster than the Jacobi method. The iterative solution is given as

$$p_{\mathbf{x}}^{(i+1,j+1)} = \frac{p_{\mathbf{x}}^{(i)} + h_i \cdot \sum_{j=1}^{R} \alpha_j(\mathbf{x}-\mathbf{v}_j) \cdot \left[\xi_{\mathbf{x}-\mathbf{v}_j} \cdot p_{\mathbf{x}-\mathbf{v}_j}^{(i+1,j+1)} + (1-\xi_{\mathbf{x}-\mathbf{v}_j}) \cdot p_{\mathbf{x}-\mathbf{v}_j}^{(i+1,j)}\right]}{1 + h_i \cdot \alpha_0(\mathbf{x})}, \quad (2.8)$$

where $\xi_{\mathbf{x}-\mathbf{v}_j}$ is an indicator (or flag) that takes the value 1 if the state $\mathbf{x} - \mathbf{v}_j$ has been already processed, and 0 otherwise. Thus, in (2.8) in the summation, we use the "new" approximations of the processed states and the "old" probability if the approximation was not yet updated in the current iteration. We modify Algorithm 4 as follows. We compute the sum as before, but after processing a state $\mathbf{x}$ in line 9, we mark it as processed and propagate $\alpha_j(\mathbf{x}) \cdot (\mathbf{x}.p_2 - \mathbf{x}.p_1)$ to the successor states which are not marked as processed. After this, the field $\mathbf{x}.k$ contains the sum required for (2.8).

## 2.4 Local error control and step size selection

The accuracy as well as the computing time of numerical integration methods depend on the order $p$ of the method and the step size. The error in a single step with step size $h$ is approximately $ch^{p+1}$ with a factor $c$ that varies over the integration interval. Hence, one crucial point for the efficiency of numerical integration methods is the step size selection. It is well known that methods with constant step size perform poorly if the solution varies rapidly in some parts of the integration interval and slowly in other parts [But08, HNW93, HW96, SGT03]. Therefore,

adaptive step size selection so that the accuracy and the computing time are well balanced is highly desirable for explicit and implicit integration schemes. For both classes of schemes we base our step size selection strategy on local error estimates.

Our goal is to control the local error and, accordingly, to choose the step size so that at each step $i$ for all states $\mathbf{x} \in Sig$,

$$|p_{\mathbf{x}}(t_i) - p_{\mathbf{x}}^{(i)}| \leq \text{RTOL} \cdot |p_{\mathbf{x}}(t_i)| + \text{ATOL}, \tag{2.9}$$

where RTOL and ATOL are user-specified relative and absolute error tolerances. In particular, note that we use a *mixed error control*, that is, a criterion that accounts for both the relative and the absolute error via corresponding relative and absolute error tolerances, because in practice using either a pure relative error control or a pure absolute error control can cause serious problems, see, e.g., Section 1.4 of [SGT03]. Of course, the true local errors are not available and we must estimate them along with each integration step.

For the explicit and the implicit Euler method we compute a local error estimate similarly to the *step doubling* approach, that is, we approximate $p^{(i+1)}$ by taking the time step $h_i$ and independently taking two consecutive time steps of length $h_i/2$. The local error estimate is then the vector

$$\hat{\varepsilon}^{(i)} = p^{(i+1),(h_i)} - p^{(i+1),(h_i/2)} \tag{2.10}$$

with components $\hat{\varepsilon}_x^{(i)}, \mathbf{x} \in Sig$, where $p^{(i+1),(h_i)}$ and $p^{(i+1),(h_i/2)}$ denote the approximations computed with time step $h_i$ and with two consecutive time steps of length $h_i/2$, respectively.

The *embedded* Runge-Kutta methods provide an alternative way for the step size control. Along with the approximation of order $p$, they deliver the approximation of order $p-1$ computed as

$$\tilde{p}^{(i+1)} = p^{(i)} + h_i \sum_{\ell=1}^{s} b_\ell^* k^{(\ell)}, \tag{2.11}$$

where $b_1^* \ldots, b_s^* > 0$ with $b_1^* + \cdots + b_s^* = 1$. Then the local error estimate is the vector

$$\hat{\varepsilon}^{(i)} = p^{(i+1)} - \tilde{p}^{(i+1)} \;=\; h_i \sum_{\ell=1}^{s} (b_\ell - b_\ell^*) k^{(\ell)} \tag{2.12}$$

with components $\hat{\varepsilon}_{\mathbf{x}}^{(i)} = p_{\mathbf{x}}^{(i+1)} - \tilde{p}_x^{(i+1)}, \mathbf{x} \in Sig$.

Now, with regard to the step size selection assume we have made a 'trial' step with a given step size $h_i$ and computed the corresponding local error estimate. Then we accept the step if for all significant states the local error estimate is smaller than the prescribed local error tolerance. More precisely,

$$\forall \mathbf{x} \in Sig : |\hat{\varepsilon}_{\mathbf{x}}^{(i)}| \leq \max(\mathrm{RTOL} \cdot \max(p_{\mathbf{x}}^{(i)}, p_{\mathbf{x}}^{(i+1)}), \mathrm{ATOL}) =: \tau_{\mathbf{x}}^{(i)}, \quad (2.13)$$

which implies

$$\forall \mathbf{x} \in Sig : |\hat{\varepsilon}_{\mathbf{x}}^{(i)}| \leq \mathrm{RTOL} \cdot \max(p_{\mathbf{x}}^{(i)}, p_{\mathbf{x}}^{(i+1)}) + \mathrm{ATOL}. \quad (2.14)$$

If the step is not accepted, then we have to decrease the step size. Otherwise we can proceed to the next step where it is likely that we can use an increased step size, because the current one might be smaller than necessary. In both cases, acceptance or rejection, we have to specify by how much the step size is decreased or increased, respectively, and in both cases we do this based on the local error estimate.

Define $\tilde{\varepsilon}_{\mathbf{x}}^{(i)} := \hat{\varepsilon}_{\mathbf{x}}^{(i)} / \tau_{\mathbf{x}}^{(i)}, \mathbf{x} \in Sig$, denote by $\tilde{\varepsilon}^{(i)}$ the corresponding vector containing the components $\tilde{\varepsilon}_{\mathbf{x}}^{(i)}$, and define

$$\alpha := \sqrt[p+1]{\frac{1}{||\tilde{\varepsilon}^{(i)}||_\infty}}. \quad (2.15)$$

It can be easily seen that the largest step size that yields a local error estimate satisfying (2.13) can be approximated by $h_i\alpha$. Note that $\alpha < 1$ if (2.13) is satisfied and $\alpha \geq 1$ otherwise. This means we can use $\alpha$ as a factor in both possible cases, that is, for a too large step size that has been rejected and must be decreased for a retrial, and for an accepted step size to set an increased step size for the next step. In practice we also have to account for the fact that the local error is only estimated. Rejecting a step and re-computing it with a smaller step size should be avoided as much as possible. Therefore, rather than $\alpha$ we consider $\rho\alpha$ with a safety factor $\rho < 1$. Besides, step sizes must not be too large and also too large changes of the step size must be avoided since otherwise the above approximation of the largest possible step size is not valid [SGT03].

If the step with step size $h_i$ is accepted, then we set

$$h_{i+1} := \min(h_{\max}, h_i \max(5, \rho\alpha)) \tag{2.16}$$

as the initial trial step size for the next step, where $h_{\max} = 0.1 \cdot T$. Otherwise, we decrease the step size for the current step according to

$$h_i := \max(h_{\min}, h_i \max(0.1, \rho\alpha)), \tag{2.17}$$

where $h_{\min} = 16 \cdot \epsilon(t_i)$ and $\epsilon(t_i)$ is the absolute distance between $t_i$ and the next floating-point number of the same precision as $t_i$. So, $h_{\min}$ is such that $t_i$ and $t_i + h_{\min}$ are different in working precision.

For the computation of the initial trial step size, we first compute $\mathcal{A}p(t_0)$ (see (1.5)). This can be done using one stage of Algorithm 3. Then we compute

$$h_0 = \max\left(h_{\min}, \min\left(h_{\max}, \rho \cdot \frac{\sqrt[p+1]{\text{RTOL}}}{\text{RTOL} \cdot \max\limits_{\mathbf{x} \in Sig} \frac{|\mathbf{x}.k_1|}{\max(\text{RTOL} \cdot \mathbf{x}.p, \text{ATOL})}}\right)\right). \tag{2.18}$$

In the $i$-th iteration, we stretch the time step $h_i$, if it lies within 10% of $T - t_i$. Thus, we set the time step $h_i$ to $(T - t_i)$ if $1.1 \cdot h_i \geq T - t_i$. Note that this also covers the case when the time step $h_i$ is too large, and using it would lead to jumping over the final time point $T$.

## 2.5 Numerical results

In this section, we present numerical examples in order to demonstrate the suitability of our approach, its accuracy, run time and the number of significant states to be processed corresponding to the number of differential equations to be integrated. As our first example we consider a birth-death process for which analytical solutions are available such that we can indeed compare our numerical results with exact values. Then we consider a more complex yeast cell polarization model.

We compare the accuracies, run times and numbers of significant states of explicit Euler (referred to as 'euler' in the following figures and tables),

implicit (backward) Euler ('beuler'), and an embedded Runge-Kutta ('rk45') with weights and coefficients chosen according to [DP80] together with local error control and adaptive step size selection as described in the previous section. Note that rk45 is similar to the `ode45` method of MATLAB, but while with MATLAB's `ode45` only systems of ODEs of moderate size can be solved, here, of course, we consider it in conjunction with our dynamical state space truncation procedure.

For our numerical experiments we fix the relative tolerance RTOL $= 10^{-3}$. For the dynamical state space truncation, we use $\delta =$ ATOL, which agrees with the error control property of the ODE solution that the components smaller than ATOL are unimportant. As a safety factor in the time step selection procedure we use $\rho = 0.8$. In the solution of the linear system required for the implicit Euler method we set $rtol =$ RTOL and $atol =$ ATOL.

### 2.5.1 Birth-death process

Our first example is the birth-death process given in Table 2.1 with $S_1$ as the only species and propensity functions $\alpha_1(\mathbf{x}) = c_1, \alpha_2(\mathbf{x}) = c_2\mathbf{x}_1$.

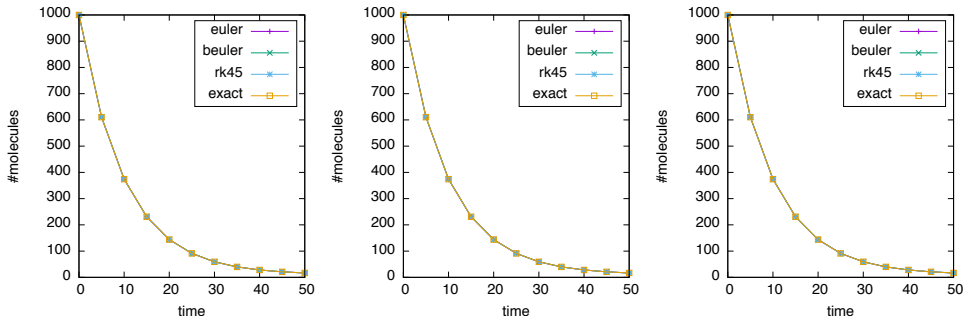
Table 2.1 – Chemical reactions of the birth-death process.

$$\emptyset \underset{c_2}{\overset{c_1}{\rightleftharpoons}} S_1$$

It is clear that the state space is the infinite set $\mathbb{N}$ of all nonnegative integers so that the corresponding system of differential equations is infinite, too. We chose the rate constants $c_1 = 1$, $c_2 = 0.1$, the initial state $\mathbf{x}_1(0) = 1000$ and final time horizon $T = 50$. We analyze the model with different values of ATOL $\in \{10^{-10}, 10^{-12}, 10^{-14}\}$. Since reporting the probabilities of single states over time is not of any practical interest we focus on representative properties and on informative measures of the accuracy and the efficiency of our method.

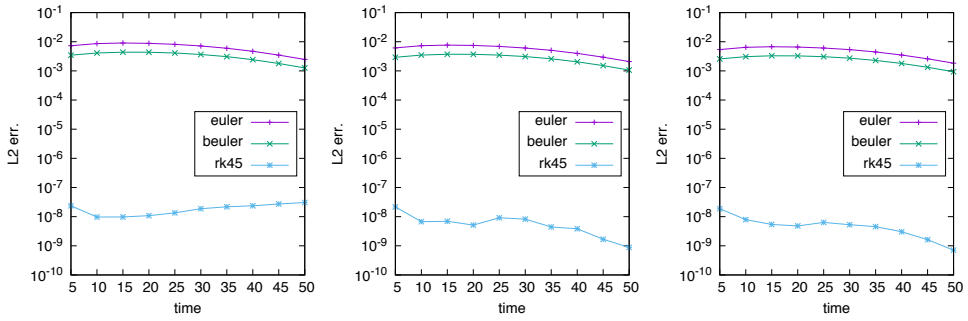
In Figure 2.2 we plot the average number of species $S_1$ over time as obtained with our approximate numerical integration schemes, where the run times were less than one second. We also plot the exact solution obtained according to [JH07]. The plots show for all considered values of ATOL that there is no visible difference between the exact values and our approximations, which suggests that our approximations are indeed

extremely accurate.




(a) (b) (c)

Figure 2.2 – Average species count for birth-death process (ATOL = (a) $10^{-10}$, (b) $10^{-12}$, (c) $10^{-14}$).




(a) (b) (c)

Figure 2.3 – L2 error for birth-death model (ATOL = (a) $10^{-10}$, (b) $10^{-12}$, (c) $10^{-14}$).

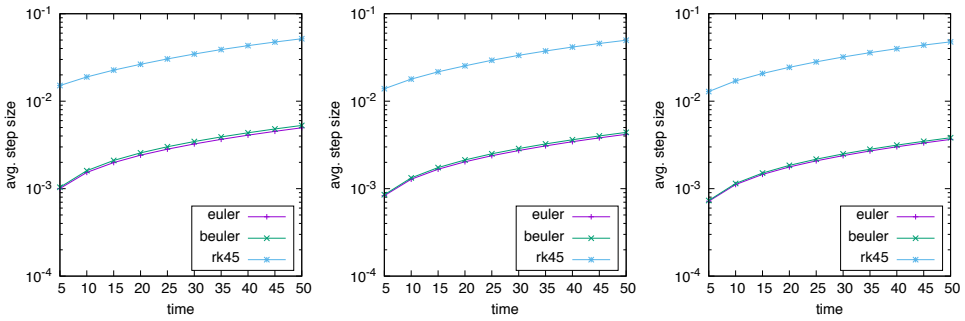
Figure 2.3 depicts plots of the L2 error

$$||p_x(t_\eta) - p_x^{(\eta)}||_2$$

with $t_\eta = T = 50$. The norm is computed over all states with positive probabilities in the exact solution. Note that if there is no corresponding state in *Sig*, its probability is taken as 0.

It can be seen that in all cases the L2 error is less than $10^{-2}$, which confirms the high accuracy of the approximations also formally. It can be further seen that for this example euler is slightly more accurate than beuler and that rk45 is even by orders of magnitude more accurate than the Euler schemes. This is well in accordance with the higher order of rk45.

Figure 2.4 shows the average step sizes taken by the different integration schemes, where rk45 takes much larger steps than the Euler schemes.



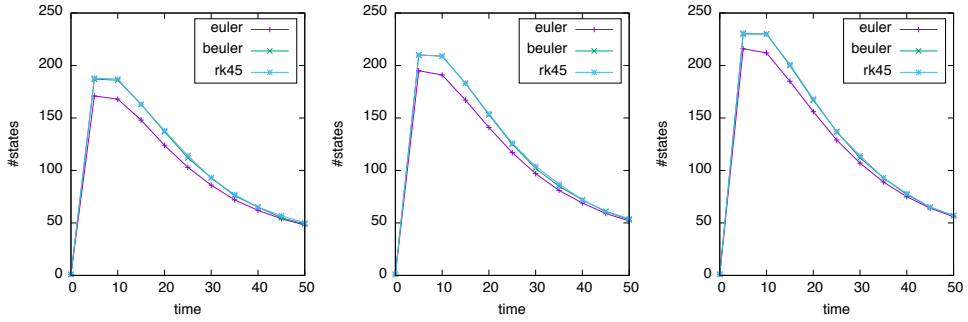

(a) (b) (c)

Figure 2.4 – Average step sizes for birth-death process (ATOL = (a) $10^{-10}$, (b) $10^{-12}$, (c) $10^{-14}$).

In Figure 2.5 we plot the numbers of significant states used during the computation. It can be seen that for all considered values of ATOL all integration schemes only require to handle (integrate) a moderate number of states (differential equations) where the Euler schemes require roughly the same number of states and rk45 requires only slightly more, in any case for any time less than 250 states. Of course, the smaller ATOL (and thus our truncation probability threshold $\delta = \text{ATOL}$) the larger the number of significant states but with only a slight increase. This shows that the dynamical truncation procedure indeed substantially reduces the size of the state space and thus renders possible to integrate numerically with – as demonstrated by the previous figures – maintaining a high accuracy of the approximations.

Table 2.2 – Chemical reactions of the yeast polarization example.

$$
\begin{aligned}
\emptyset &\overset{c_1}{\underset{c_2}{\rightleftharpoons}} R \\
L + R &\xrightarrow{c_3} RL + L \\
RL &\xrightarrow{c_4} R \\
RL + G &\xrightarrow{c_5} G_a + G_{bg} \\
G_a &\xrightarrow{c_6} G_d \\
G_d + G_{bg} &\xrightarrow{c_7} G \\
\emptyset &\xrightarrow{c_8} RL
\end{aligned}
$$




(a) (b) (c)

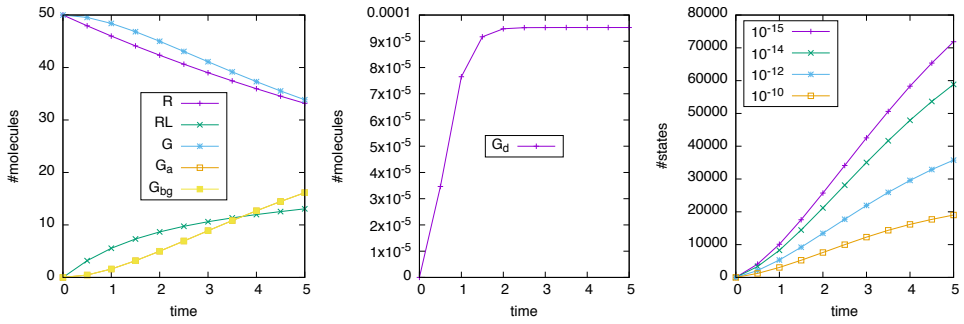
Figure 2.5 – Numbers of significant states for birth-death process (ATOL = (a) $10^{-10}$, (b) $10^{-12}$, (c) $10^{-14}$).

### 2.5.2 Yeast cell polarization

As another reference example we consider a stochastic model of the pheromone-induced G-protein cycle in the yeast Saccharomyces cerevisiae [CNY08, PB00, RDJGP11] given in Table 2.2. We chose the reaction rate constants $\mathbf{c} = (0.0038, 0.0004, 0.042, 0.01, 0.011, 0.1, 1050.0, 3.21)$ and the initial state $\mathbf{x}(0) = (50, 2, 0, 50, 0, 0, 0)$, where the state vector is given as $\mathbf{x} = (R, L, RL, G, G_a, G_{bg}, G_d)$ and the state space is the infinite 7-dimensional set $\mathbb{N}^7$. Note that in order to keep the meaning of the species here we do not number the species but take the notation from [RDJGP11].

In Figure 2.6 we plot the average species counts computed using rk45 with ATOL $= 10^{-15}$ and the numbers of significant states for rk45 over time for different values of ATOL $\in \{10^{-10}, 10^{-12}, 10^{-14}, 10^{-15}\}$. Note that as in the previous example the numbers of significant states as well as the average numbers of species for euler and beuler only slightly differ from those for rk45, so that we omit to include them in the plots. It is clear that in the much more complex yeast cell polarization model there are many more significant states than in the birth-death process, but the numbers of significant states are still in a range that allows accurate approximations in reasonable time.

In Table 2.3 we list the run times for different values of ATOL. We can see that in this example beuler heavily outperforms the explicit methods euler and rk45. This confirms the advantages of implicit methods over explicit methods for stiff systems. In fact, while the birth-death example is not




(a) (b) (c)

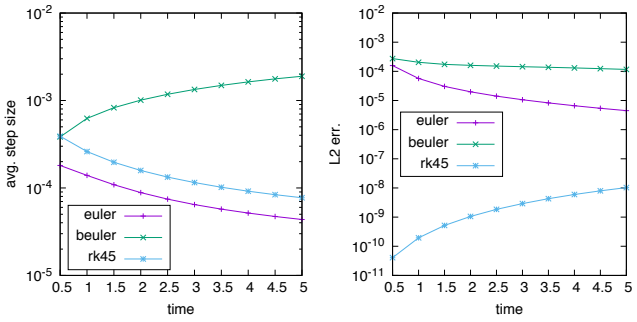
Figure 2.6 – Yeast cell polarization: (a,b) average species counts, (c) number of significant states for different values of ATOL.

or only moderately stiff, the yeast cell polarization model constitutes a very stiff system of differential equations.

Table 2.3 – Run times for yeast cell polarization model.

| method | ATOL $= 10^{-10}$ | ATOL $= 10^{-12}$ | ATOL $= 10^{-14}$ |
|---|---|---|---|
| euler | 761s | 1481s | 5839s |
| beuler | 35s | 76s | 139s |
| rk45 | 5806s | 18603s | 26126s |

In Figure 2.7 we plot the average step size over time and the L2 error for ATOL $= 10^{-14}$. Since there is no analytical solution, we compute the L2 with respect to the distribution obtained using rk45 with a lower absolute tolerance ATOL $= 10^{-15}$, applying the rationale that with an even lower error tolerance the method gives nearly exact values. The respective L2 errors for ATOL $= 10^{-10}$ and ATOL $= 10^{-12}$ are similar.




(a) (b)

Figure 2.7 – (a) Average step sizes and (b) L2 error for yeast polarization model (ATOL = $10^{-14}$).

# 3 Hybrid Numerical Solution of the CME

The analysis of stochastic models that describe networks of chemical reactions is either based on Monte Carlo sampling or on a numerical solution of the underlying chemical master equation (CME). The latter is advantageous for the estimation of parameters since likelihoods and their derivatives can be computed simultaneously [AMSW11]. Numerical methods, however, become slow when too many states have to be considered. If some species are present in high copy numbers even neglecting low-probability states is not enough to make a numerical solution feasible.

As an orthogonal approach, moment-based analysis methods have been developed [Eng06, Hes08, RMASL11, LKK09]. Instead of integrating the distribution of all states over time, the idea is to represent the distribution by its (multidimensional) moments up to order $M$ and integrate the moments over time. The number of equations is, even for, say, $M = 10$, very small compared to the number of states and thus the system of equations that has to be integrated is very small in comparison to the CME. In addition, the number of moments does not increase as the copy number of certain species increases.

Moment-based approaches, however, have a number of disadvantages. First of all, one has to reconstruct the distribution from the moments at the final time point of the integration, which is a difficult optimization problem that can only be solved for small dimensions [AMW15b]. Then, the moment equations often become very stiff, in particular for $M > 4$. Moreover, the generation of the correct moment equations is difficult due to the combinatorial nature of the multidimensional moments and thus it is non-trivial to implement such approaches. Another problem

is that the moment representation of certain systems looses information about qualitative properties such as oscillation and multistability. For instance, the stochastic version of the predator-prey model shows damped oscillations of the predator means for certain parameter combinations. However, with $M = 2$ the moment equations are unable to yield a correct solution [DMW10]. For $M > 2$ the equations result in a system that is singular up to working precision.

Hybrid methods have been developed to mitigate the difficulties of moment-based approaches and ensure scalability as certain chemical populations become large [HWKT13, HMMW10, MLSH12, Jah11]. The basic idea is to integrate the marginal distributions of low copy-number species according to a "small" master equation and couple this equation with the conditional moment equations for the large copy-number populations. This is a natural representation in particular for gene regulatory networks since low copy-number species represent the state of a gene (active or inactive) and thus the moment equations are integrated for all different possible gene states. The resulting system of differential equations is typically less susceptible to numerical instabilities and for increasing $M$ the total size of the system is smaller than the size of the system in a purely moment-based approach. Also, for increasing $M$ hybrid methods show greater accuracy for all moments of order $< M$ compared to the moments obtained from a purely moment-based approach [HWKT13].

Hasenauer et al. presented the method of conditional moments (MCM) where the system of differential equations for the conditional moments and the small master equation were derived for an arbitrary truncation order $M$ [HWKT13, KTH14]. When the probability of the condition is zero this system contains algebraic equations and thus requires the use of a solver for differential algebraic equations (DAE). In addition, Hasenauer et al. discuss in detail that transformations to ordinary differential equations by setting such probabilities to a small value $\epsilon$ may result in large approximation errors.

In this chapter we show how the equations obtained from the MCM can be solved by a simple integration of the differential equations without the need of solving algebraic equations. In addition, we combine the MCM with state truncation approaches that have been proposed for the numerical solution of the CME in Chapter 2. In this way, it is possible to condition on species with an unbounded copy-number which is useful

for any chemical population that has a small mean and is therefore a driver for the stochasticity of the system. As shown by Hasenauer et al., for such species a moment representation may be inadequate and a static truncation was used to keep the population range finite. Here, we use a dynamic truncation of the range that neglects population numbers with a probability smaller than a truncation threshold $\delta$. We describe in detail, how the MCM can be combined with a state truncation and how the solution of algebraic equations can be avoided without a significantly larger approximation error.

## 3.1 Stochastic hybrid modeling of biochemical reaction networks

A numerical solution of the CME is usually extremely slow or infeasible for large models even when using truncation-based methods, because a very large number of states has to be processed [HMMW10, AMSW11, AMSW12, MW12]. An alternative is to use a hybrid state space representation, where some species are represented by their distribution whilst others are represented by their moments. Formally, the set of species is split into two sets $\mathbb{S}^{(y)}$ and $\mathbb{S}^{(z)}$ ($\mathbb{S} = \mathbb{S}^{(y)} \cup \mathbb{S}^{(z)}$). We consider the process $\mathbf{X}(t) = (\mathbf{Y}(t)\ \mathbf{Z}(t))$, where $\mathbf{Y}(t)$ describes the evolution of the discrete species $\mathbb{S}^{(y)}$ over time and $\mathbf{Z}(t)$ describes the evolution of the continuous species $\mathbb{S}^{(z)}$ over time. For each reaction $j$, the change vectors $\mathbf{v}_j^{(y),-}$, $\mathbf{v}_j^{(z),-}$ and $\mathbf{v}_j^{(y),+}$, $\mathbf{v}_j^{(z),+}$ are then the corresponding entries of $\mathbf{v}_j^-$ and $\mathbf{v}_j^+$ respectively.

We denote the probability distribution of $\mathbf{Y}(t)$ as

$$p(\mathbf{y}|t) = Pr(\mathbf{Y}(t) = \mathbf{y})\,.$$

Note that

$$p(\mathbf{x}|t) = p(\mathbf{z}|\mathbf{y}, t)p(\mathbf{y}|t).$$

We denote the conditional non-central moments of $\mathbf{Z}(t)$ as

$$\mu_{\mathbf{I},\mathbf{z}}(\mathbf{y}, t) = \mathbb{E}_z[\mathbf{Z}^{\mathbf{I}}(t)|\mathbf{y}, t],$$

and the conditional central moments as

$$M_{\mathbf{I},\mathbf{z}}(\mathbf{y},t) = \mathbb{E}_z[(\mathbf{Z}(t) - \mu_z(\mathbf{y},t))^{\mathbf{I}}|\mathbf{y},t],$$

where $\mathbf{I} = (I_1,\ldots,I_{N_z}), I_i \geq 0$ and $\mathbf{Z}^{\mathbf{I}}(t) = \prod_{i=1}^{N_z} Z_i^{I_i}(t)$.

### 3.1.1 Method of conditional moments

In [KTH14], the equations for the evolution of the marginal probabilities $p(\mathbf{y}|t)$ and the conditional moments $\mu_{\mathbf{I},\mathbf{z}}(\mathbf{y},t)$ are derived for a given partitioning $\mathbb{S}^{(y)} \cup \mathbb{S}^{(z)}$. The corresponding system of DAEs is given by

$$\begin{aligned}\frac{d}{dt}p(\mathbf{y}|t) = &- p(\mathbf{y}|t)\sum_{j=1}^{R}\mathbb{E}_z[\alpha_j(\mathbf{y},\mathbf{Z})|\mathbf{y},t] \\ &+ \sum_{\substack{j=1 \\ \mathbf{y}\geq\mathbf{v}_j^{(y),+}}}^{R} p(\mathbf{y}-\mathbf{v}_j^{(y)}|t)\mathbb{E}_z[\alpha_j(\mathbf{y}-\mathbf{v}_j^{(y)},\mathbf{Z})|\mathbf{y}-\mathbf{v}_j^{(y)},t]\end{aligned} \tag{3.1}$$

$$\begin{aligned}&p(\mathbf{y}|t)\frac{d}{dt}\mu_{\mathbf{I},\mathbf{z}}(\mathbf{y},t) + \mu_{\mathbf{I},\mathbf{z}}(\mathbf{y},t)\frac{d}{dt}p(\mathbf{y}|t) = \\ &- p(\mathbf{y}|t)\sum_{j=1}^{R}\mathbb{E}_z[\alpha_j(\mathbf{y},\mathbf{Z})\mathbf{Z}^{\mathbf{I}}|\mathbf{y},t] \;+ \sum_{\substack{j=1 \\ \mathbf{y}\geq\mathbf{v}_j^{(y),+}}}^{R} p(\mathbf{y}-\mathbf{v}_j^{(y)}|t) \\ &\qquad\times\mathbb{E}_z[\alpha_j(\mathbf{y}-\mathbf{v}_j^{(y)},\mathbf{Z})(\mathbf{Z}+\mathbf{v}_j^{(z)})^{\mathbf{I}}|\mathbf{y}-\mathbf{v}_j^{(y)},t]\end{aligned} \tag{3.2}$$

Equation (3.1) is a "small" master equation for the marginal distributions of low copy-number species and Equation (3.2) describes the evolution of the conditional moments of the large copy-number populations. Equations (3.1) and (3.2) contain the expectation $\mathbb{E}_z[\alpha_j(\mathbf{y},\mathbf{Z})\mathbf{Z}^{\mathbf{I}}|\mathbf{y},t]$, which can be evaluated using a Taylor series expansion (TSE) of $\alpha_j(\mathbf{y},\mathbf{Z})\mathbf{Z}^{\mathbf{I}}$ about the mean:

$$\mathbb{E}_z[\alpha_j(\mathbf{y},\mathbf{Z})\mathbf{Z}^{\mathbf{I}}|\mathbf{y},t] = \sum_{I_1=0}^{\infty}\cdots\sum_{I_{N_z}=0}^{\infty}\frac{1}{\mathbf{I}!}\frac{\partial^{\mathbf{I}}\left(\alpha_j(\mathbf{y},\mathbf{z})\mathbf{z}^{\mathbf{I}}\right)}{\partial\mathbf{z}^{\mathbf{I}}}\bigg|_{\mathbf{z}=\mu_z(\mathbf{y},t)} M_{\mathbf{I},\mathbf{z}}(\mathbf{y},t), \tag{3.3}$$

where $\partial^{\mathbf{I}} = \partial^{I_1+\ldots+I_{N_z}}$, $\partial\mathbf{z}^{\mathbf{I}} = \partial z_1^{I_1}\ldots\partial z_{N_z}^{I_{N_z}}$ and $\mathbf{I}! = I_1!\ldots I_{N_z}!$.

Since the propensity functions are not restricted to be polynomial, their

TSEs are infinite and have to be truncated. We use the low dispersion closure scheme setting the central moments of order greater than $M$ to zero, i.e. $M_{\mathbf{I},\mathbf{z}}(\mathbf{y},t) = 0$ if $\sum_{i=1}^{N_z} I_i > M$.

In contrast to [HWKT13, KTH14], our implementation operates on a dynamical state space which includes only states *Sig* with significant probability mass ($> \delta$) (see Section 2.1). Therefore, the term $p(\mathbf{y}|t)\frac{d}{dt}\mu_{\mathbf{I},\mathbf{z}}(\mathbf{y},t)$ in (3.2) doesn't become zero. Thus, the DAEs (3.2) can be rewritten as ODEs for the partial conditional moments $\mu^*_{\mathbf{I},\mathbf{z}}(\mathbf{y},t) = p(\mathbf{y}|t)\mu_{\mathbf{I},\mathbf{z}}(\mathbf{y},t)$. The right hand hand side of (3.1) and (3.2) contains $\mu_{\mathbf{I},\mathbf{z}}(\mathbf{y},t)$ which can be computed by dividing $\mu^*_{\mathbf{I},\mathbf{z}}(\mathbf{y},t)$ by $p(\mathbf{y}|t)$ because $p(\mathbf{y}|t) > 0$. For newly added state $\mathbf{y}$ to *Sig*, the partial conditional moments $\mu^*_{\mathbf{I},\mathbf{z}}(\mathbf{y},t)$ are initialized with zero because the state probability is assumed to be 0.

### 3.1.2 Example: dimerisation

We illustrate the conditional moment equations using a simple dimerisation system [Wil11], which includes two reactions given in Table 3.1. Note that the propensity functions for the first and the second reaction are given as $c_1\frac{x_P(x_P-1)}{2}$ and $c_2 x_{P_2}$ respectively, where $x_P$ is the concentrations of $P$ and $x_{P_2}$ is the concentrations of $P_2$. We set $M = 2$ and consider partitionings of the set of species $\mathbb{S}^{(y)} = \{P, P_2\}$, $\mathbb{S}^{(z)} = \emptyset$ and $\mathbb{S}^{(y)} = \{P\}$, $\mathbb{S}^{(z)} = \{P_2\}$.

Table 3.1 – Chemical reactions of the simple dimerisation system [Wil11].

$$2\ \mathrm{P} \quad \underset{c_2}{\overset{c_1}{\rightleftharpoons}} \quad \mathrm{P}_2$$

1. $\mathbb{S}^{(y)} = \{\mathrm{P}, \mathrm{P}_2\}$, $\mathbb{S}^{(z)} = \emptyset$

$$\begin{aligned}\frac{d}{dt}p((x_P, x_{P_2}),t) = &- p((x_P, x_{P_2}),t)\left(\frac{c_1}{2}x_P(x_P-1) + c_2 x_{P_2}\right)\\ &+ \left[p((x_P+2, x_{P_2}-1),t)\frac{c_1}{2}(x_P+2)(x_P+1)\right]_{x_{P_2}\geq 1}\\ &+ \left[p((x_P-2, x_{P_2}+1),t)c_2(x_{P_2}+1)\right]_{x_P\geq 2}\end{aligned}$$

2. $\mathbb{S}^{(y)} = \{\mathrm{P}\}$, $\mathbb{S}^{(z)} = \{\mathrm{P}_2\}$

$$\begin{aligned}
\frac{d}{dt}p(x_P,t) =& -p(x_P,t)\left(\frac{c_1}{2}x_P(x_P-1)+c_2\mu_{P_2}(x_P,t)\right)\\
&+\ p(x_P+2,t)\frac{c_1}{2}(x_P+2)(x_P+1)\\
&+\ [p(x_P-2,t)c_2\mu_{P_2}(x_P-2,t)]_{x_P\geq 2}\\
\frac{d}{dt}\mu^*_{P_2}(x_P,t) =& -p(x_P,t)\left(\frac{c_1}{2}x_P(x_P-1)\mu_{P_2}(x_P,t)+c_2\mu_{P_2^2}(x_P,t)\right)\\
&+\ p(x_P+2,t)\frac{c_1}{2}(x_P+2)(x_P+1)\mu_{P_2}(x_{P+2},t)\\
&+\ \left[p(x_P-2,t)c_2\left(\mu_{P_2^2}(x_P-2,t)+\mu_{P_2}(x_P-2,t)\right)\right]_{x_P\geq 2}\\
\frac{d}{dt}\mu^*_{P_2^2}(x_P,t) =& -p(x_P,t)\left(\frac{c_1}{2}x_P(x_P-1)\mu_{P_2^2}(x_P,t)+c_2\mu_{P_2^3}(x_P,t)\right)\\
&+\ p(x_P+2,t)\frac{c_1}{2}(x_P+2)(x_P+1)\mu_{P_2^2}(x_{P+2},t)\\
&+\ \Big[p(x_P-2,t)c_2\Big(\mu_{P_2^3}(x_P-2,t)+2\mu_{P_2^2}(x_P-2,t)\\
&\qquad +\ \mu_{P_2}(x_P-2,t))\Big]_{x_P\geq 2}
\end{aligned}$$

Similarly, the systems of ODEs (3.1) and (3.2) can be derived for partitionings of the set of species $\mathbb{S}^{(y)} = \{P_2\}$, $\mathbb{S}^{(z)} = \{P\}$ and $\mathbb{S}^{(y)} = \emptyset, \mathbb{S}^{(z)} = \{P, P_2\}$.

## 3.2 Numerical algorithm

In this section we present the implementation details of the solution of the system of ODEs, which results from the system of DAEs (3.1)-(3.2) under the assumption that the state probabilities $p(\mathbf{y}|t)$ are greater than zero. Our algorithm for the hybrid solution of the CME is based on the numerical adaptive step size integraton scheme described in Chapter 2. Alongside with the state probabilities, for each state $\mathbf{y}$ we store the partial conditional moments $\mu^*_{\mathbf{I},\mathbf{z}}(\mathbf{y})$.

We consider $k$-dimensional tetrahedral arrays $\mu^{*,k}_{\mathbf{A},z}(\mathbf{y})$ with $\mathbf{A} = [a_1, \ldots, a_k]$

and $1 \leq a_k \leq \ldots \leq a_2 \leq a_1 \leq N_z$ $(k = 1 \ldots M)$ such that

$$\mu^{*,k}_{\mathbf{A},z}(\mathbf{y}) = \mu^{*}_{\mathbf{I},z}(\mathbf{y}) \iff \forall i \in 1, \ldots, N_z : I_i = \sum_{j=1}^{k} \mathbb{1}_{a_j=i}, \tag{3.4}$$

where $\mathbb{1}_{a_j=i}$ equals 1 if and only if $a_j = i$ and equals 0 otherwise. Thus, we write out indices of species in lexicographic order and each index is written as many times as the number in $\mathbf{I}$ vector. For example, if $\mathbf{I} = (2, 3, 1)$, then $\mathbf{A} = [1, 1, 2, 2, 2, 3]$.

Then, for each state $\mathbf{y} \in \mathit{Sig}$, the marginal probability $p(\mathbf{y})$ and the conditional partial moments $\mu^{*}_{z}(\mathbf{y})$ can be stored in a vector $\mathbf{u}(\mathbf{y}) = (p(\mathbf{y}),$ $\mu^{*,1}_{z}(\mathbf{y}), \mu^{*,2}_{z}(\mathbf{y}), \ldots, \mu^{*,M}_{z}(\mathbf{y}))$. Thus, we can rewrite the system of ODEs as:

$$\frac{d}{dt}\mathbf{u}(t) = F\left(\mathbf{u}\right), \tag{3.5}$$

where $F$ corresponds to the right hand side of the modified system of equations (3.1) and (3.2).

We denote the number of elements in the array $\mu^{*,k}_{z}$ as $\mathrm{len}(\mu^{*,k}_{z})$. Also, we denote the indices of $p$ and $\mu^{*}_{\mathbf{I},\mathbf{z}}$ in the vector $\mathbf{u}$ as $\mathrm{loc}(p)$ and $\mathrm{loc}(\mu^{*}_{\mathbf{I},\mathbf{z}})$ respectively. Then we have the following equalities:

$$\mathrm{len}(\mu^{*,k}_{z}) = \binom{N_z + k - 1}{k},$$

$$\mathrm{loc}(p) = 0,$$

$$\mathrm{loc}(\mu^{*}_{i,\mathbf{z}}) = 1 + i,$$

$$\mathrm{loc}(\mu^{*}_{\mathbf{I},\mathbf{z}}) = \mathrm{loc}(\mu^{*,k}_{\mathbf{A},z}) = 1 + \sum_{j=1}^{k-1} \mathrm{len}(\mu^{*,j}_{z}) + \sum_{r=1}^{k} \binom{a_r + k - r}{1 + k - r}.$$

The number of partial conditional moments for a single state $\mathbf{y}$ can be computed as

$$\sum_{j=1}^{M} \mathrm{len}(\mu^{*,j}_{z}) = \binom{M + N_z}{M} - 1,$$

and several values are given in Table 3.2. Thus, the total number of

equations in the system of ODEs is equal to

$$|Sig| \cdot \binom{M + N_z}{N_z},$$

which is the number of states with significant probability times the number of equations per state.

| $N_z$ \ $M$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 | 2 | 5 | 9 | 14 | 20 | 27 | 35 | 44 |
| 3 | 3 | 9 | 19 | 34 | 55 | 83 | 119 | 164 |
| 4 | 4 | 14 | 34 | 69 | 125 | 209 | 329 | 494 |
| 5 | 5 | 20 | 55 | 125 | 251 | 461 | 791 | 1286 |
| 6 | 6 | 27 | 83 | 209 | 461 | 923 | 1715 | 3002 |
| 7 | 7 | 35 | 119 | 329 | 791 | 1715 | 3431 | 6434 |
| 8 | 8 | 44 | 164 | 494 | 1286 | 3002 | 6434 | 12869 |

Table 3.2 – Number of partial conditional moments for a single state $\mathbf{y}$ for varying values of $N_z$ and $M$.

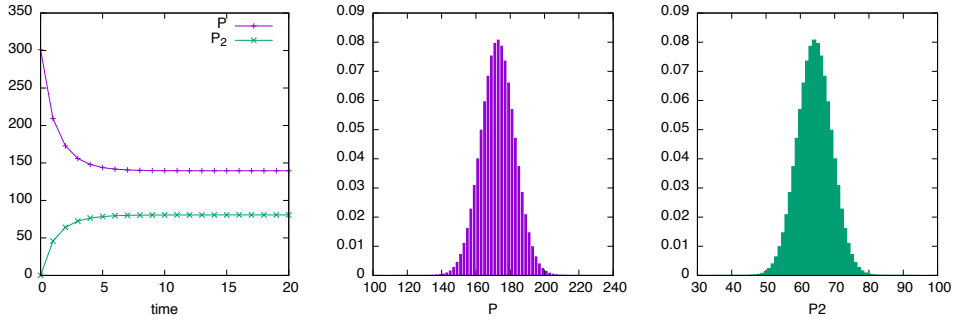
## 3.3 Numerical results

In this section we investigate two biological systems, with polynomial as well as non-polynomial propensity functions. We apply our algorithm of conditional moments using different partitionings of species and compare the accuracy and the running time. In the following, we denote the method of conditional moments for $M$ moments and for the set of discrete species $\mathbb{S}^{(y)}$ as $\mathrm{MCM}_{\mathrm{M}}\left(\mathbb{S}^{(y)}\right)$. Note that $\mathrm{MCM}_{/}(\mathbb{S})$ corresponds to the full stochastic solution.

### 3.3.1 Dimerisation

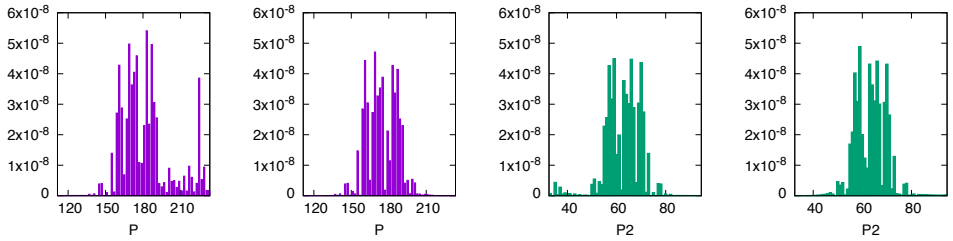
The first example is the dimerisation model from Section 3.1.2 with the parameters $\mathbf{c} = (1.66 \cdot 10^{-3}, 0.2)$ and the initial counts $\mathbf{x}_0 = (301, 0)$. In Figure 3.1(a) we plot the means of P and $\mathrm{P}_2$ obtained using $\mathrm{MCM}_{/}(\mathbb{S})$. In Figure 3.1(b,c) we plot their probability distributions for $t = 2$.

Further, we consider $\mathrm{MCM}_1(\mathrm{P})$ and $\mathrm{MCM}_2(\mathrm{P})$ and plot the absolute




(a) (b) (c)

Figure 3.1 – Results of the full stochastic solution of the dimerisation example: a) means of P and $P_2$; b,c) probability distribution of P and $P_2$ respectively at $t = 2$.



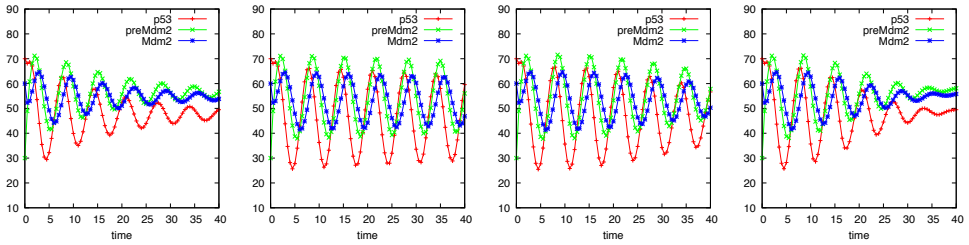

(a) (b) (c) (d)

Figure 3.2 – Absolute difference between the marginal distrubution of P (a,b) and $P_2$ (c,d) computed using the fully stochastic numerical approach and the distribution computed using the hybrid approach $\mathrm{MCM}_{1\ldots2}(\mathrm{P})$ and $\mathrm{MCM}_{1\ldots2}(\mathrm{P}_2)$ respectively at $t = 2$.

difference between the marginal distrubution of P computed using the fully stochastic numerical approach and the distribution computed using the hybrid approach $\mathrm{MCM}_{1\ldots2}(\mathrm{P})$. It can be seen that the accuracy slightly improves at the edge of the distribution when increasing $M$ from 1 to 2. Similarly, we plot results obtaind for $\mathrm{MCM}_{1\ldots2}(\mathrm{P}_2)$ in Figure 3.2(c,d).

### 3.3.2 P53 system

The second example is the oscillatory p53 system, consisting of three proteins p53, precursor of Mdm2 and Mdm2, which are connected via a nonlinear feedback loop [AKS13]. The reactions of the system are given in Table 3.3 where $x_1$, $x_2$ and $x_3$ are the concentrations of p53, preMdm2




(a) (b) (c) (d)

Figure 3.3 – Means of p53, preMdm2 and Mdm2 computed using (a) SSA, 1000000 trajectories; (b) $\mathrm{MCM}_1\,(\emptyset)$; (c) $\mathrm{MCM}_4\,(\emptyset)$; (d) $\mathrm{MCM}_6\,(\emptyset)$.

and Mdm2 respectively, and $\alpha_3 = c_3 \frac{x_3 x_1}{x_1 + c_7}$, $\alpha_4 = c_4 x_1$. The parameters of the system are chosen as $\mathbf{c} = (90.0, 0.002, 1.7, 1.1, 0.93, 0.96, 0.01)$ and the initial counts are $\mathbf{x}_0 = (70, 30, 60)$. In Figure 3.5 (a) we plot the means of the protein concentrations computed using 1000000 SSA trajectories.

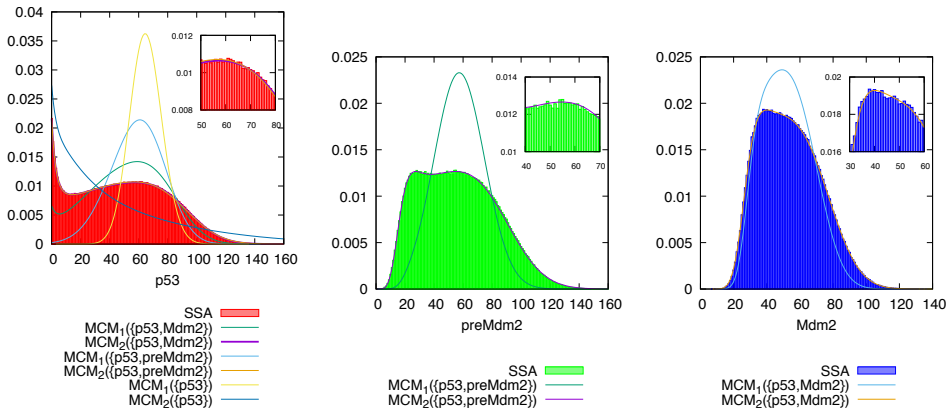
Table 3.3 – Chemical reactions of the p53 system [AKS13].

$$
\begin{aligned}
\emptyset &\underset{c_2}{\overset{c_1}{\rightleftharpoons}} \text{p53} \\
\text{p53} &\xrightarrow{*} \emptyset \\
\emptyset &\xrightarrow{*} \text{pMdm2} \\
\text{pMdm2} &\xrightarrow{c_5} \text{Mdm2} \\
\text{Mdm2} &\xrightarrow{c_6} \emptyset
\end{aligned}
$$

Firstly, we consider the case when $\mathbb{S}^{(y)} = \emptyset$. We plot the means of the protein concentrations in Figure 3.5 (b), (c), (d) for $M = 1$, $M = 4$ and $M = 6$ respectively. The accuracy of the method clearly becomes higher as the number of moments $M$ increases. For $M > 6$ we were not able to obtain the results due to too small time step during the solution of the system of ODEs.

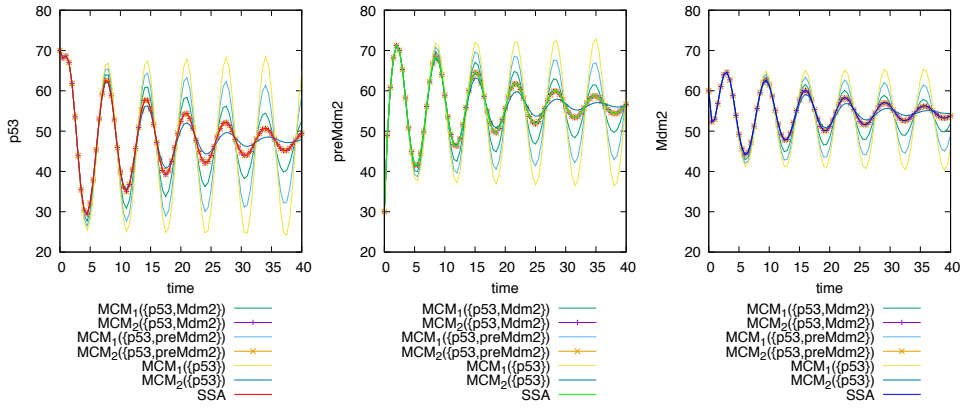
Further, we consider the case when $\mathbb{S}^{(y)} = \{\text{p53}\}$ and $M = 1, 2$ which resulted in a very inaccurate solution (see Figures 3.4, 3.5). Choosing $M > 2$ or $\mathbb{S}^{(y)}$ containing a protein other than p53, resulted in an impossibility of solving the system of ODEs.

Finally, we consider cases when $\mathbb{S}^{(y)} = \{\text{p53}, \text{preMdm2}\}$ and $\mathbb{S}^{(y)} = \{\text{p53}, \text{Mdm2}\}$ with $M = 1, 2$. For $M = 1$ the obtained solution was inaccurate (see Figures 3.4, 3.5). For $M = 2$ the solution is very accurate compared to the solution obtained using SSA. The computation time was 87 hours for $\mathbb{S}^{(y)} = \{\text{p53}, \text{preMdm2}\}$ and 83 hours for $\mathbb{S}^{(y)} = \{\text{p53}, \text{Mdm2}\}$.




(a) (b) (c)

Figure 3.4 – Probability distribution of (a) p53, (b) preMdm2 and (c) Mdm2 computed using different approaches.




(a) (b) (c)

Figure 3.5 – Means of (a) p53, (b) preMdm2 and (c) Mdm2 computed using different approaches.

# 4 Numerical Approximation of Rare Event Probabilities

In many cases, rare events, that is events that occur with a very small probability, are particularly important, for instance because they describe a system behavior of high practical relevance or because they may have serious consequences. Examples include population sizes exceeding an exceptionally high level or falling below an exceptionally low level during some fixed time period, extinction of molecular species, outbreak of infectious diseases, apoptosis (cell death), or rare but important transitions between different long-lived stable regions in metastable systems, amongst many others. Determining the probabilities of such rare but important events is highly desirable.

Explicit closed-form solutions of the CME are usually not available such that it has to be solved numerically. However, the size of the multi-dimensional state space of the underlying CTMC typically increases exponentially with the number of molecular species, hence with the model dimensionality. This effect is known as state space explosion and often causes models to be numerically intractable due to the prohibitively large, often even infinite state space.

Therefore, the most widespread approach to analyzing stochastic chemical kinetics is stochastic simulation [Gil76, Gil77], which means to mimic the time evolution of a biochemically reacting system by repeatedly generating trajectories (sample paths) of the underlying CTMC with the help of computer-generated random numbers. Mathematically, this constitutes a statistical estimation procedure for system properties such as expectations, moments and cumulants of molecular population sizes, or probabilities of certain events of interests. Proper statistical output analysis yields point

estimators and confidence intervals [AG07, KTB13, San09b].

Stochastic simulation does not suffer from state space explosion because the state space need not be explicitly enumerated, but stochastic simulation tends to be computationally expensive and can only provide estimates whose reliability and accuracy in terms of relative errors or confidence interval half widths depend on the variance of the corresponding simulation estimator. In particular, estimating rare event probabilities by 'standard' simulation is inefficient, because rare events are simulated too infrequently. The variance and the relative error of the corresponding standard estimators are much too large to obtain statistically reliable estimates in reasonable time. Variance reduction and specific rare event simulation techniques are required [AG07, Buc13, RT$^+$09].

Importance sampling is a variance reduction technique that makes use of a change of measure. The probability distribution (measure) in the model is changed such that the rare event of interest becomes less rare. Then the simulation is conducted under the importance sampling measure and the systematically biased results are weighted by the likelihood ratio in order to yield unbiased estimates. However, importance sampling by no means guarantees variance reduction but may be even counterproductive and increase the variance. The efficiency of corresponding simulation schemes and the statistical accuracy of simulation results, usually expressed by asymptotic robustness properties of the underlying importance sampling estimators [LBTG10, San07, TLS06], strongly depend on the choice of the change of measure.

Despite recent progress in the application of such techniques to biochemically reacting systems [San09a, KM08, GRP09, DJRGP11, RDJGP11], as outlined above a clear disadvantage of stochastic simulation compared to numerical analysis, provided that such an analysis would be possible, is the inherent statistical uncertainty of simulation results, that is estimates of the probabilities of interest. Thus, we argue that if a problem may be tackled both by stochastic simulation and by numerical analysis, the latter should be preferred.

In this chapter, we consider a numerical solution approach for the computation of rare event probabilities. It combines the basic idea of importance sampling with a numerical solution approach that overcomes the state space explosion by using a state space truncation described in Chap-

ter 2.1. The underlying principle is a guided state space exploration where paths that contribute significantly to the rare event probability are not truncated. We use parameter biasing strategies similarly as in rare event simulation to identify the significant parts of the state space and 'guide' the exploration of the state space in such a way that an accurate approximation of the rare event probability is obtained.

## 4.1 Importance sampling

Importance sampling aims at variance reduction for simulation estimators by a change of measure. The original system is simulated under a different probability measure and weighting by a correcting factor, the likelihood ratio, yields unbiased estimates. In a general measure theoretic setting, importance sampling is based on the Radon-Nikodym theorem, and all applications of importance sampling can be derived from this setting.

Consider two probability measures $P$ and $Q$ on a measurable space $(\Omega, \mathcal{A})$, where $P$ is absolutely continuous with respect to $Q$, that is $\forall A \in \mathcal{A} : Q(A) = 0 \Rightarrow P(A) = 0$, or equivalently, $\forall A \in \mathcal{A} : P(A) > 0 \Rightarrow Q(A) > 0$. Then, the Radon-Nikodym theorem guarantees the existence of the Radon-Nikodym derivative $L = dP/dQ$, often also referred to as the likelihood ratio, and

$$\forall A \in \mathcal{A} : \; P(A) = \int_A L dQ. \tag{4.1}$$

Importance sampling basically exploits that expectations with respect to $P$ are identical to expectations with respect to $Q$ when weighting by the likelihood ratio. Hence, for random variables $X$ on $(\Omega, \mathcal{A})$,

$$E_P[X] = \int X dP = \int X L dQ = E_Q[XL]. \tag{4.2}$$

The probability of an event $A \in \mathcal{A}$ can be expressed as a special case by $P(A) = E_P[I_A]$ where $I_A$ denotes the indicator function of $A$.

For CTMCs, the relevant probability measures are path distributions and absolute continuity corresponds to the condition that all paths that are possible under the original measure must remain possible under the importance sampling measure. This can be obviously achieved by the condition that for all positive rates in the original model the corresponding

rates under importance sampling are positive. Since we deal with CTMCs given in terms of transition classes as described in Section 1.2, we need an appropriate framework for the application of importance sampling to this model specification.

With importance sampling, the underlying probability measure determined by the transition rate functions is changed. Since the only requirement is absolute continuity of the probability measures involved, there is much freedom in how to change the measure. It is only necessary that all paths that are possible (have positive probability) under the original measure remain possible. This can be achieved by changing the original transition rate functions $\alpha_i$ to 'importance sampling transition rate function' $\tilde{\alpha}_i$ such that for all $i \in \{1, \ldots, R\}$ we have $\tilde{\alpha}_i(\mathbf{x}) = 0 \;\Rightarrow\; \alpha_i(\mathbf{x}) = 0,\ \mathbf{x} \in \mathcal{X}$, or equivalently, starting with the original propensity functions, $\alpha_i(\mathbf{x}) > 0 \;\Rightarrow\; \tilde{\alpha}_i(\mathbf{x}) > 0,\ \mathbf{x} \in \mathcal{X}$. One then generates trajectories according to the changed transition rate functions and multiplies the results with the likelihood ratio to get unbiased estimates for the original system. The trajectory generation now yields a sequence of states with associated sojourn times and reaction path density as in (1.7). If we set $\tilde{\alpha}_0(\mathbf{x}(t_{i-1})) := \tilde{\alpha}_1(\mathbf{x}(t_{i-1})) + \cdots + \tilde{\alpha}_R(\mathbf{x}(t_{i-1}))$ and keep the same initial distribution at time $t_0$ as for the original model, the likelihood ratio of a trajectory $\omega$ is

$$L(\omega) = \prod_{i=1}^{K} \frac{\alpha_{j_i}\left(\mathbf{x}(t_{i-1})\right) \exp\left(-\alpha_0(\mathbf{x}(t_{i-1}))\tau_{i-1}\right)}{\tilde{\alpha}_{j_i}\left(\mathbf{x}(t_{i-1})\right) \exp\left(-\tilde{\alpha}_0(\mathbf{x}(t_{i-1}))\tau_{i-1}\right)} \tag{4.3}$$

which can be efficiently computed during trajectory generation without much extra computational effort by successively updating its value after each simulated reaction according to the running product.

However, the results possess variances and are thus subject to statistical uncertainty since stochastic simulation yet with importance sampling still remains an estimation procedure. If the change of measure is chosen properly, it yields enormous variance reduction compared to direct simulation, but as a serious drawback of importance sampling a badly chosen change of measure can even lead to infinite variance increase. Moreover, in practice, the true probability as well as the unknown variance of the estimator must be estimated in course of the simulation and both are often significantly underestimated, which then leads to wrong conclusions and much too narrow confidence intervals that may even not contain the

rare event probability of interest. Hence, also the reliability of importance sampling simulation results is extremely sensitive to the change of measure.

Nevertheless, the change of measure is an advantageous strategy for systematically increasing the probability of certain events and thus provides useful hints how to guide the system under study in order to provoke rare events of interest. We shall therefore exploit it in order to efficiently compute rare event probabilities without resorting to stochastic simulation, thereby avoiding both the statistical uncertainty inherent in simulation results and the danger of accidentally neglecting relevant parts of the state space as often the case with conventional state space truncation procedures.

## 4.2 Guided state space exploration

For many practical applications, the accuracy of the approximation method in Chapter 2 is sufficient for a moderately small choice of the truncation thresholds $\delta$ and $\hat{\delta}$, respectively. If, however, the probabilities of rare events have to be calculated, then the truncation approach is no longer appropriate. As it stands now, the main drawback of the truncation approach is that rare events of interest may be neglected, that is, the truncated state space may not include those paths that lead to a certain rare event because their probability is smaller than the corresponding truncation threshold. If smaller truncation thresholds are chosen then the paths that significantly contribute to the rare event probability may not be truncated, but the number of states that have to be considered may become too large to be manageable.

In this section we propose an extension of the truncation approach presented in Chapter 2 that is inspired by ideas from importance sampling and recent weighted stochastic simulation algorithms for estimating rare event probabilities [San09a, KM08, GRP09, DJRGP11, RDJGP11]. Assume that we are interested in the probability $P(A)$ of a rare event $A$. Besides the CTMC $(\mathbf{X}(t))_{t\geq 0}$, we consider another CTMC $(\tilde{\mathbf{X}}(t))_{t\geq 0}$ with the same state space and the same reaction channels but with different propensity functions $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_R$ instead of the true propensity functions $\alpha_1, \ldots, \alpha_R$. We choose these 'biased' propensity functions $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_R$ such that the occurence of $A$ is more likely than with $\alpha_1, \ldots, \alpha_R$. Then we use

**Algorithm 5** A modification of Algorithm 3 for the computation of rare event probabilities using the guidance functions $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_R$.

**for** $\ell \leftarrow 1$ to $s$ **do**
  **for all** $\mathbf{x} \in \mathit{Sig}$ **do**
    $\hat{p} \leftarrow x.p + h \cdot \sum_{j=1}^{\ell-1} a_{\ell j} \cdot \mathbf{x}.k_j, \quad \hat{q} \leftarrow x.q + h \cdot \sum_{j=1}^{\ell-1} a_{\ell j} \cdot \mathbf{x}.m_j$
    **for all** $j \in \{1, \ldots, R\} : x + \mathbf{v}_j \geq \mathbf{0}$ **and** $\left(\{\mathbf{x} + \mathbf{v}_j\} \in \mathit{Sig}$ **or** $h \cdot \tilde{\alpha}_j(\mathbf{x}) \cdot \hat{q} > \hat{\delta}\right)$ **do**
      **if** $\{\mathbf{x} + \mathbf{v}_j\} \notin \mathit{Sig}$ **then** $\mathit{Sig} \leftarrow \mathit{Sig} \cup \{\mathbf{x} + \mathbf{v}_j\}$ **end if**
      $\mathbf{x}.k_\ell \leftarrow \mathbf{x}.k_\ell - h \cdot \alpha_j(\mathbf{x}) \cdot \hat{p}, \quad \mathbf{x}.m_\ell \leftarrow \mathbf{x}.m_\ell - h \cdot \tilde{\alpha}_j(\mathbf{x}) \cdot \hat{q}$
      $(\mathbf{x} + \mathbf{v}_j).k_\ell \leftarrow (\mathbf{x} + \mathbf{v}_j).k_\ell + h \cdot \alpha_j(\mathbf{x}) \cdot \hat{p}, \quad (\mathbf{x} + \mathbf{v}_j).m_\ell \leftarrow (\mathbf{x} + \mathbf{v}_j).m_\ell + h \cdot \tilde{\alpha}_j(\mathbf{x}) \cdot \hat{q}$
    **end for**
  **end for**
**end for**
**for all** $\mathbf{x} \in \mathit{Sig}$ **do**
  $\mathbf{x}.p_2 \leftarrow \mathbf{x}.p + h \cdot \sum_{j=1}^{s} b_j \cdot \mathbf{x}.k_j, \quad \mathbf{x}.q_2 \leftarrow \mathbf{x}.q + h \cdot \sum_{j=1}^{s} b_j \cdot \mathbf{x}.m_j$
  $\mathbf{x}.k_1 \leftarrow 0, \ldots, \mathbf{x}.k_s \leftarrow 0, \quad \mathbf{x}.m_1 \leftarrow 0, \ldots, \mathbf{x}.m_s \leftarrow 0$
**end for**

the CTMC $(\tilde{\mathbf{X}}(t))_{t \geq 0}$ as 'guide' with regard to the state space truncation, that is essentially the propensity functions $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_R$ guide us through the state space exploration in such a way that paths to the rare event of interest are not truncated. Therefore, we refer to $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_R$ as guidance functions.

The idea is to solve $\mathbf{X}$ and $\tilde{\mathbf{X}}$ simultaneously using the dynamical state space truncation. Let $\hat{\mathbf{p}}^{(t)}$ ($\hat{\mathbf{q}}^{(t)}$) be the corresponding numerical approximation of the distribution of $\mathbf{X}$ (of $\tilde{\mathbf{X}}$) at time $t$, respectively. The algorithm for solving $\mathbf{X}$ and $\tilde{\mathbf{X}}$ (Algorithm 5) is a modification of Algorithm 3. Instead of a single field $\mathbf{x}.p$ for the current probability of state $\mathbf{x}$ we use two fields $\mathbf{x}.p$ and $\mathbf{x}.q$. The former refers to the current probability of state $\mathbf{x}$ in $\mathbf{X}$ and the latter refers to the probability of $\mathbf{x}$ in $\tilde{\mathbf{X}}$. The decision whether we remove a state $\mathbf{x}$ from the set *Sig* at time $t$ depends only on $\hat{\mathbf{q}}^{(t)}$ and not on $\hat{\mathbf{p}}^{(t)}$. Thus, at all time instances $t$ for both the solution of $\mathbf{X}$ and $\tilde{\mathbf{X}}$ we use the same sets *Sig*. This ensures that we do not truncate the paths leading to the rare event $A$. Intuitively, $\tilde{\mathbf{X}}$ shows the direction to the rare event. Therefore, we refer to this approach as guided state space exploration. If the guidance functions are chosen appropriately, then the vectors $\hat{\mathbf{q}}^{(t)}$ are computed using those paths that contribute most to $P(A)$. Hence, the vector $\hat{\mathbf{p}}^{(t)}$ may loose a lot of probability mass over time, that is $\sum_{\mathbf{x} \in S} \hat{p}^{(t)}(\mathbf{x}) \ll 1$. The probability

mass that remains in $\hat{\mathbf{p}}^{(t)}$ then contains those parts that contribute most to $P(A)$.

Note that the rare event probability $P(A)$ is directly approximated by the probabilities $\mathbf{x}.p$ and the values $\mathbf{x}.q$ are only used to determine the set of states that are considered in each step of the numerical integration. Actually, it would even be possible to solve $\tilde{\mathbf{X}}$ and $\mathbf{X}$ not simultaneously but one after another. During the solution of $\tilde{\mathbf{X}}$, we would then record the elements of *Sig* for each time interval and use this information for the subsequent solution of $\mathbf{X}$ during which we truncate the state space in the same way as for $\tilde{\mathbf{X}}$. The simultaneous solution, however, has the advantage that it is faster than two subsequent solutions.

Let $P_\delta(A)$ denote the computed approximation with the approach outlined above where $\delta$ is the chosen significance threshold. It is important to note that, if we ignore errors of the numerical integration method, it holds that

$$P_\delta(A) \leq P(A) \tag{4.4}$$

because some paths leading to $A$ may be truncated. Moreover, as $\delta \to 0$ our approximation approaches $P(A)$, that is, $\lim_{\delta \to 0} P_\delta(A) = P(A)$. In particular, when we decrease $\delta$ the accuracy will improve. Thus, if we apply the guided state space exploration for decreasing values of $\delta$ and see that $P_\delta(A)$ converges, we can estimate the approximation error as $P_\delta(A) - P_{\tilde{\delta}}(A)$ where $\delta < \tilde{\delta}$.

## 4.3 Numerical results

In this section, we present experimental results for specific rare event probabilities computed by the guided state space exploration described above. We consider certain buffer overflow probabilities corresponding to high numbers of customers in single nodes or in the overall system, respectively, for two variants of a two-node tandem Jackson network and an eight-node tandem Jackson network. The two-node networks are standard examples from the literature for which appropriate changes of measure have been studied extensively in the context of importance sampling. The eight-node network is a more complex example for which appropriate changes of measure for importance sampling have not been

investigated yet. As a benchmark example we also consider the enzymatic futile cycle.

In the sequel, we present results where we systematically vary the parameters determining the change of measure to study the sensitivity of our approach and consider different values for the truncation threshold $\delta$. We ran our experiments on an Intel Core i7 at 2.8 GHz with 8 GB main memory. As an integration method we used the standard explicit fourth-order Runge-Kutta method with the time step chosen as the smallest average sojourn time of all states in *Sig*, that is,

$$h = \min_{\mathbf{x} \in Sig} 1/\sum_{j=1}^{R} \alpha_j(\mathbf{x}).$$

### 4.3.1 Tandem Jackson networks

As a queueing network example consider a $N$-node tandem Jackson network with exponentially distributed service times where arrivals occur only at the first node according to a Poisson process with arrival rate $\lambda$. The service rates are denoted by $\mu_1, \ldots, \mu_N$ and the buffer capacities (queue sizes) by $\nu_1, \ldots, \nu_N$. Hence, the different types of transitions are arrivals at node 1, moves from node $i$ to node $i+1$, $0 < i < N$ and departures from node $N$. Therefore, $N+1$ transition classes are sufficient:

$\mathcal{C}_1 = (\mathcal{U}_1, u_1, \alpha_1)$, where

- $\mathcal{U}_1 = \{(x_1, \ldots, x_N) \in \mathbb{N}^N : x_1 < \nu_1\}$,
- $u_1(\mathbf{x}) = (x_1 + 1, x_2, x_3, \ldots, x_N)$,
- $\alpha_1(\mathbf{x}) = \lambda$;

$\mathcal{C}_i = (\mathcal{U}_i, u_i, \alpha_i), i = 2, \ldots, N$, where

- $\mathcal{U}_i = \{(x_1, \ldots, x_N) \in \mathbb{N}^N : x_{i-1} > 0, x_i < \nu_i\}$,
- $u_(\mathbf{x}) = (x_1, \ldots, x_{i-2}, x_{i-1} - 1, x_i + 1, x_{i+1}, \ldots, x_N)$,
- $\alpha_i(\mathbf{x}) = \mu_{i-1}$;

$\mathcal{C}_{N+1} = (\mathcal{U}_{N+1}, u_{N+1}, \alpha_{N+1})$, where

- $\mathcal{U}_{N+1} = \{(x_1, \ldots, x_N) \in \mathbb{N}^N : x_N > 0\}$,
- $u_{N+1}(\mathbf{x}) = (x_1, \ldots, x_{N-1}, x_N - 1)$,
- $\alpha_{d+1}(\mathbf{x}) = \mu_N$.

State-dependent rates can be easily incorporated just by corresponding transition rate functions. The state space may be infinite due to one or more infinite buffers, which can be expressed explicitly by setting the buffer size to infinity or implicitly just by dropping the corresponding restrictions on the respective source state spaces. Phase-type distributed interarrival and service times can be modeled by properly defined transition classes for any change from one to the next phase.

**Two-node tandem network** Our first example is a two-node tandem Jackson network with infinite buffers at both nodes, hence a special case where now $N = 2$ and $\nu_1 = \nu_2 = \infty$. The arrival and service rates are constant whereby we can skip the state dependence of the transition rate functions and concisely express them by a triplet $\alpha := (\alpha_1, \alpha_2, \alpha_3) := (\lambda, \mu_1, \mu_2)$. Similarly, we express the changed measure by $\tilde{\alpha} = (\tilde{\alpha}_1, \tilde{\alpha}_2, \tilde{\alpha}_3)$ where $\tilde{\alpha}_1$ is the changed arrival rate and $\tilde{\alpha}_2, \tilde{\alpha}_3$ are the changed service rates at nodes 1 and 2, respectively. We are interested in the probability that the overall population in the system reaches a level $L$ during a busy cycle, that is, the probability that starting with an arrival to the empty system the sum of the number of customers in both network nodes is at least $L$ before the system empties again. Obviously, for low utilizations and/or high "target level" $L$, reaching $L$ during a busy cycle is a rare event.

Table 4.1 – Exact solution results for the two-node tandem network with parameters $\lambda = 0.04$, $\mu_1 = \mu_2 = 0.48$.

| $L$ | Pr | $\lvert Sig \rvert$ |
|---|---|---|
| 12 | 1.4693e-11 | 90 |
| 25 | 2.8722e-25 | 350 |
| 50 | 6.0327e-52 | 1325 |

Though at a first glance seemingly simple, this example has received a lot of attention in the rare event simulation literature and has become a major reference example for judging change of measure strategies. This enormous interest was basically initiated by a change of measure proposed in [PW89] where the arrival rate and the smaller service rate (or, respectively, the service rate at the second node in case of equal service rates) are exchanged. Though initially supposed to be efficient, this change of measure has been subsequently proven in [GK95] to perform poorly in certain critical

Table 4.2 – Guided state space exploration results for the two-node tandem network with parameters $\lambda = 0.04$, $\mu_1 = \mu_2 = 0.48$. Six different changes of measure considered: 1. degenerated, 2. $0.48, 0.48, 0.04$, 3. $0.6222, 0.3333$, $0.0444$, 4. $0.4, 0.3, 0.3$, 5. $0.6, 0.2, 0.2$, 6. $0.8, 0.1, 0.1$.

| | $\delta$ | L=12 | | L=25 | | L=50 | |
|---|---|---|---|---|---|---|---|
| | | rel.err. | $\lvert Sig\rvert$ | rel.err. | $\lvert Sig\rvert$ | rel.err. | $\lvert Sig\rvert$ |
| 1 | 1e-20 | 3.461e-8 | 89 | 1 | 184 | 1 | 184 |
| | 1e-15 | 4.881e-3 | 88 | 1 | 103 | 1 | 103 |
| | 1e-10 | 1 | 54 | 1 | 54 | 1 | 54 |
| 2 | 1e-20 | 3.408e-14 | 58 | 1.288e-9 | 235 | 5.349e-4 | 757 |
| | 1e-15 | 1.339e-9 | 46 | 1.257e-5 | 185 | 6.756e-2 | 591 |
| | 1e-10 | 3.397e-5 | 34 | 1.817e-3 | 132 | 4.220e-1 | 398 |
| 3 | 1e-20 | 2.012e-11 | 47 | 3.777e-9 | 192 | 4.104e-6 | 650 |
| | 1e-15 | 4.589e-8 | 38 | 4.186e-6 | 155 | 1.149e-3 | 525 |
| | 1e-10 | 8.166e-5 | 30 | 2.488e-3 | 116 | 9.555e-2 | 383 |
| 4 | 1e-20 | 0 | 89 | 7.034e-15 | 343 | 2.746e-8 | 1259 |
| | 1e-15 | 0 | 89 | 1.403e-9 | 342 | 5.277e-4 | 1233 |
| | 1e-10 | 6.338e-9 | 89 | 1.552e-4 | 337 | 1.902e-1 | 1177 |
| 5 | 1e-20 | 0 | 77 | 7.034e-15 | 337 | 1.245e-7 | 1145 |
| | 1e-15 | 6.817e-15 | 60 | 1.374e-9 | 269 | 7.538e-3 | 877 |
| | 1e-10 | 3.632e-8 | 42 | 4.987e-4 | 180 | 2.628e-1 | 524 |
| 6 | 1e-20 | 7.892e-8 | 39 | 1.361e-4 | 155 | 2.225e-1 | 444 |
| | 1e-15 | 3.171e-5 | 33 | 3.519e-2 | 121 | 3.960e-1 | 309 |
| | 1e-10 | 1.082e-2 | 25 | 3.241e-1 | 75 | 6.209e-1 | 178 |

parameter (arrival and service rates) regions. A recent thorough analysis can be found in [DB06]. As the example has been so extensively studied in the context of importance sampling it enables us to demonstrate the advantages of our algorithm over importance sampling. For numerical analysis, we choose the parameter setting $\alpha = (0.04, 0.48, 0.48)$, hence arrival rate $\lambda = 0.04$ and service rates $\mu_1 = \mu_2 = 0.48$, which belongs to the critical parameter regions ascertained in [GK95]. We consider three different levels $L \in \{12, 25, 50\}$ and six different changes of measure $\tilde{\alpha}^{(1)}, \ldots, \tilde{\alpha}^{(6)}$ as follows.

First of all, we keep the original rates, that is, we do not at all apply a change of measure in this case, which we grasp as the "degenerated change of measure" $\tilde{\alpha}^{(1)} = (0.04, 0.48, 0.48)$. The second change of measure is

the interchange of the arrival rate and the service rate at the second node according to [PW89], hence $\tilde{\alpha}^{(2)} = (0.48, 0.48, 0.04)$. The third one is $\tilde{\alpha}^{(3)} = (0.6222, 0.3333, 0.0444)$ developed in [San04a] and shown to provide better results than $\tilde{\alpha}^{(2)}$ when used for importance sampling. Furthermore, we consider $\tilde{\alpha}^{(4)} = (0.4, 0.4, 0.3)$, $\tilde{\alpha}^{(5)} = (0.6, 0.2, 0.2)$ and $\beta^{(6)} = (0.8, 0.1, 0.1)$, none of which developed for rare event simulation with importance sampling but rather ad hoc chosen by us. They are simply based on the intuitive reasoning that increasing the arrival rate and decreasing the service rate obviously guides the tandem network to a higher population level and thus increases the rare event probability of interest. Since these ad hoc changes of measure do not yield proper importance sampling simulation results, they are particularly well suited to highlight that our algorithm is far less sensitive to the change of measure than importance sampling and that in contrast to importance sampling our algorithm does not require intricate pre-analyses.

Table 4.3 – Exact solution results for the two-node tandem network with parameters $B = 100$ and $\lambda = 0.1$, $\mu_1 = 0.7$, $\mu_2 = 0.2$.

| Pr | run time | $\lvert Sig \rvert$ |
|---|---|---|
| 9.2034e-31 | 87s | 36774 |

In Table 4.1 we list the rare event probabilities (cf. column "Pr") as well as the size of the set *Sig* of significant states obtained using the truncation threshold $\delta = 0$. The rare event probabilities computed in this way are exact up to the numerical integration error. We list our guided state space exploration results in Table 4.2 for different values of $\delta$ (listed in the second column) whereas the first column contains the index of the change of measure. For all parameters that we chose, the running time of our algorithm is less than one second.

The six different changes of measures are indicated in the table by their respective parameter values except for the degenerated change of measure that is indicated by "degenerated". For each change of measure the column with heading "rel.err." lists the approximation error relative to the rare event probability.

For the degenerated change of measure, the guided state space exploration is identical to the dynamical state space truncation presented in Chapter 2 (but in contrast to the exact solution with positive truncation threshold $\delta$) and it yields accurate approximations only for $L = 12$

and $\delta \in \{10^{-15}, 10^{-20}\}$. Note that a relative error of one corresponds to approximating the rare event probability as zero. This is actually the same as what happens in direct simulation when due to the small probability the rare event is not observed and the probability of the non-observed event is estimated as zero. It is important that with our dynamical state space exploration the relative error is bounded by one since any error in the approximation is due to neglecting relevant states. In contrast, with importance sampling the relative error expressed in terms of the estimator's coefficient of variation or the relative half width of the corresponding confidence interval can become arbitrarily large in cases where the rare event has been observed but the estimator's variance is large (cf. [DB06, GK95, San04a]).

For $\tilde{\alpha}^{(2)}$ according to [PW89] and $\tilde{\alpha}^{(3)}$ according to [San04a] we can see that for all levels and all truncation thresholds the results provided by the guided state space exploration are very accurate. In [San04a] it is shown that with importance sampling $\tilde{\alpha}^{(2)}$ performs very poor for high levels in that for $L = 25$ it yields results with a relative error of nearly 800%, whereas $\tilde{\alpha}^{(3)}$ still yields statistically accurate results. For $L = 50$, neither $\tilde{\alpha}^{(2)}$ nor $\tilde{\alpha}^{(3)}$ yield useful results with importance sampling. Here, with the guided state space exploration both changes of measure perform extremely well even for $L = 50$ where $\tilde{\alpha}^{(3)}$ is only slightly better $\tilde{\alpha}^{(2)}$. For the latter case, the relative error is at most of the order of $10^{-4}$ if $\delta = 10^{-20}$ while it is at most of the order of $10^{-6}$ in the former case. Moreover, for $\tilde{\alpha}^{(2)}$ the set *Sig* is slightly larger than for $\tilde{\alpha}^{(3)}$. Hence, altogether both changes of measure perform quite similarly, which clearly indicates that the guided state space exploration is less sensitive to the change of measure, or, in other words, the impact of the specific change of measure on the reliability of the results is far lower.

For the ad hoc changes of measure $\tilde{\alpha}^{(4)}$ and $\tilde{\alpha}^{(5)}$ we observe that our results are again very accurate for small enough truncation threshold, though less accurate than the results for $\tilde{\alpha}^{(2)}$ and $\tilde{\alpha}^{(3)}$. However, only if $\delta$ is too large the relative error becomes high. Since neither of these changes of measure properly works with importance sampling, they are particularly well suited to further corroborate and highlight again that our algorithm is far less sensitive to the change of measure than importance sampling and that in contrast to importance sampling our algorithm does not require intricate pre-analyses.

Finally with the last case, $\tilde{\alpha}^{(6)} = (0.8, 0.1, 0.1)$, we test a change of measure that has disastrous effects when used with importance sampling. It is beyond our scope here to provide a detailed analysis of this change of measure in the context of importance sampling but some explanation of the effects can be given by the notion of overbiasing. Overbiasing is a well known problem in importance sampling and basically means too much simulation acceleration. Although the goal is to provoke more of the rare events of interest in order to reduce the variance of the simulation estimator, it can be shown that provoking too many of them yields the contrary effect. More formally, overbiasing in importance sampling yields extremely small likelihood ratios for most of the corresponding simulation runs but very large likelihood ratios for a few simulation runs. This results in an enormous variance of the importance sampling estimators since this variance is mainly driven by the variance of the likelihood ratio. This effect is actually one of the main causes for the extreme sensitivity of importance sampling to the change of measure.

As we can see from our results in Table 4.2 the overbiasing effects seems to play a role also for the guided state space exploration but it is much less serious than for importance sampling. Our algorithm, though less efficient than for other changes of measure, still provides proper results for reasonably small truncation thresholds. Once again we get evidence that our algorithm is far less sensitive to the change of measure than importance sampling.

In Tables 4.3, 4.4, we list further results for the two-node tandem network but other than before we now do not consider the level of the overall population but the probability that the second queue reaches some high buffer content $B$. For the original network, we consider the parameter combinations studied in [MSM07], namely $\lambda = 0.1$, $\mu_1 = 0.7$, $\mu_2 = 0.2$, and we compute the probability that the second queue contains at least $B = 100$ customers. In contrast to the previous type of rare event probabilities as presented in Table 4.2, we solve an infinite system without an indirectly given bound since now the number of customers in the first queue is not any longer implicitly bounded by a target level of the overall population. In addition to the relative error and the average size of *Sig*, we list the run time of our method (cf. columns "run time"). The exact solution takes 87 seconds while a solution with the change of measure $(0.2, 0.7, 0.1)$ takes only a few seconds. The degenerated change of measure yields a relative error of one while the other heuristically chosen changes,

Table 4.4 – Guided state space exploration results for the two-node tandem network with parameters $B = 100$ and $\lambda = 0.1$, $\mu_1 = 0.7$, $\mu_2 = 0.2$. Four different changes of measure considered: 1. degenerated, 2. $(0.2, 0.7, 0.1)$, 3. $(0.4, 0.4, 0.2)$, 4. $(0.3, 0.6, 0.1)$.

| | $\delta$ | rel.err. | run time | $\lvert Sig\rvert$ |
|---|---|---|---|---|
| 1 | 1e-20 | 1 | 1s | 348 |
| | 1e-15 | 1 | $< 1s$ | 198 |
| | 1e-10 | 1 | $< 1s$ | 93 |
| 2 | 1e-20 | 1.1419e-15 | 7s | 1769 |
| | 1e-15 | 1.2178e-10 | 4s | 1022 |
| | 1e-10 | 8.9310e-6 | 2s | 456 |
| 3 | 1e-20 | 2.8028e-5 | 42s | 8008 |
| | 1e-15 | 2.8139e-3 | 22s | 4561 |
| | 1e-10 | 1.3226e-1 | 8s | 1974 |
| 4 | 1e-20 | 4.0291e-3 | 6s | 1198 |
| | 1e-15 | 5.3733e-2 | 3s | 707 |
| | 1e-10 | 4.0188e-1 | 2s | 325 |

$(0.4, 0.4, 0.2)$ and $(0.3, 0.6, 0.1)$, yield good results except if $\delta$ is chosen too large.

**Two-node tandem network with slow-down** Our next example network is a slight modification of the previous one, the two-node tandem Jackson network with server slow-down as considered in, e.g., [DLW07, MSM07, Mir09]. When the length of the second queue reaches $B \cdot \theta$, the service rate of the first node changes from $\mu_1$ to $\nu_1$. Similar to the example above, the network with slow-down has become a reference example for judging change of measure strategies for systems with state-dependent rates in the original system. Here, we consider two different parameter combinations from [MSM07]. The first combination is $(\lambda, \mu_1, \mu_2, \nu_1) = (0.1, 0.7, 0.2, 0.3)$ and the second one is $(\lambda, \mu_1, \mu_2, \nu_1) = (0.1, 0.7, 0.2, 0.15)$ where $\lambda$ is the arrival rate, $\mu_2$ is the service rate at the second node, and $\mu_1$ and $\nu_1$ are the service rates at the first node before and after slow-down. We compute the probability that starting with an arrival to the empty system the number of customers in the second network node is at least $B$ before the system empties again.

We list the exact results for both cases in Table 4.5. We list the guided

Table 4.5 – Exact solution results for the two-node tandem network with slow-down and parameters $B = 100$, $\theta = 0.8$. Parameter set 1: $\lambda = 0.1$, $\mu_1 = 0.7$, $\mu_2 = 0.2$, $\nu_1 = 0.3$, parameter set 2: $\lambda = 0.1$, $\mu_1 = 0.7$, $\mu_2 = 0.2$, $\nu_1 = 0.15$.

| | Pr | run time | $\lvert Sig\rvert$ |
|---|---|---|---|
| 1 | 5.6009e-31 | 142s | 54049 |
| 2 | 3.5471e-32 | 172s | 71841 |

state space exploration results in Table 4.6 where the first column refers to the change of measure. Also, in both cases we let $B = 100$ and $\theta = 0.8$. Similar as above, we consider an exact solution ($\delta = 0$) and the degenerated change of measure. The changes of measure suggested in [MSM07] are such that, for pcom 1, arrival rate $\mu_2$ is chosen, service rate at the second node is $\lambda$, and the service rate at the first node is chosen as $\mu_1$ while the queue length is below $B \cdot \theta$ and $\nu_1$ if the queue length is at least $B \cdot \theta$ (cf. column "as suggested in [MSM07]"). For pcom 2, the parameters of the change of measure are also chosen as $(\mu_2, \mu_1, \lambda)$ while the queue length at the first queue is below $B \cdot \theta$. If the queue length reaches $B \cdot \theta$, the parameters of the change of measure are calculated by solving an equation (see [MSM07]) which yields $(0.177263, 0.177263, 0.095474)$. Again, we list the results of this change of measure in the column with heading "as suggested in [MSM07]". Finally, we consider for both parameter combinations two heuristically chosen changes of measure, $(0.4, 0.4, 0.2, \nu_1)$ and $(0.3, 0.6, 0.1, \nu_1)$ where $\nu_1 = 0.3$ for pcom 1 and $\nu_1 = 0.15$ for pcom 2. Similar as for the network without slowdown, the exact solution takes significantly longer than the solutions based on the dynamical truncation of the state space. Moreover, the relative error is one for the degenerated case and high if $\delta$ is large and the change of measure is chosen heuristically. The change of measure suggested in [MSM07] performs well even if $\delta$ is large.

**Eight-node tandem network** Our final example is a tandem network with eight nodes. We choose arrival rate $\lambda = 0.04$, and equal service rates $\mu_1 = \ldots = \mu_8 = 0.12$ and consider the probability that starting with an arrival to the empty system the sum of the number of customers in all network nodes is at least $L = 29$ before time $T$. Hence, as in our very first example we are concerned with the overall population in the system but we restrict our analysis to the time interval $[0, T]$ where $T = 50$ or

Table 4.6 – Guided state space exploration results for the two-node tandem network with slow-down and parameters $B = 100$, $\theta = 0.8$. Parameter sets are as in Table 4.5. Four different changes of measure considered: 1. degenerated, 2. as suggested in [MSM07], 3. $(0.4, 0.4, 0.2, \nu_1)$, 4. $(0.3, 0.6, 0.1, \nu_1)$.

| | $\delta$ | 1 | | | 2 | | |
|---|---|---|---|---|---|---|---|
| | | rel.err. | run time | $\lvert Sig\rvert$ | rel.err. | run time | $\lvert Sig\rvert$ |
| 1 | 1e-20 | 1 | $< 1s$ | 348 | 1 | $< 1s$ | 344 |
| | 1e-15 | 1 | $< 1s$ | 198 | 1 | $< 1s$ | 197 |
| | 1e-10 | 1 | $< 1s$ | 93 | 1 | $< 1s$ | 93 |
| 2 | 1e-20 | 1.8764e-15 | 14s | 2534 | 2.7777e-15 | 32s | 5164 |
| | 1e-15 | 1.0977e-10 | 8s | 1444 | 2.6431e-10 | 21s | 3464 |
| | 1e-10 | 7.9622e-6 | 4s | 631 | 4.9194e-5 | 11s | 1851 |
| 3 | 1e-20 | 3.9418e-5 | 81s | 13730 | 3.1021e-3 | 228s | 30590 |
| | 1e-15 | 3.8105e-3 | 40s | 7009 | 1.2373e-1 | 150s | 20777 |
| | 1e-10 | 1.6369e-1 | 14s | 2699 | 9.0224e-1 | 81s | 11450 |
| 4 | 1e-20 | 4.1024e-3 | 11s | 1886 | 1.1562e-2 | 70s | 10364 |
| | 1e-15 | 5.4456e-2 | 6s | 1089 | 1.2141e-1 | 44s | 6590 |
| | 1e-10 | 4.0330e-1 | 3s | 497 | 6.2285e-1 | 17s | 2480 |

Table 4.7 – Exact solution results for the eight-node tandem network.

| $T$ | Pr | run time | $\lvert Sig\rvert$ |
|---|---|---|---|
| 50 | 1.6643e-23 | 796s | 36984860 |
| 100 | 1.2204e-16 | 2317s | 36984860 |

$T = 100$ (cf. column "$T$"). In Table 4.7 we list the probabilities $P(A)$ for two different choices of $T$ as well as the run time and the size of the set $Sig$ of significant states obtained by setting $\delta = 0$.

To the best of our knowledge, no in-depth study of this system has been conducted in the context of importance sampling. As it is more complex than the two-node networks, of course, an appropriate change of measure is more difficult to obtain. Here, we consider heuristic changes of measure that seem reasonable with regard to the goal of provoking more rare events of interest. As before, we start by considering the degenerated change of measure where no rates are changed and the computation is done using the parameters of the original network. For the non-degenerated changes of measure we first apply a quite straightforward generalization of exchanging the arrival rate with the service rate at the last node, hence

Table 4.8 – Guided state space exploration results for the eight-node tandem network. Four changes of measures considered: 1. degenerated, 2. $\lambda$ and $\mu_8$ exchanged, 3. $\lambda$ two times faster, 4. $\lambda$ three times faster.

| | $\delta$ | $T = 50$ | | | $T = 100$ | | |
|---|---|---|---|---|---|---|---|
| | | rel.err. | run time | $\lvert Sig \rvert$ | rel.err. | run time | $\lvert Sig \rvert$ |
| 1 | 1e-20 | 1 | 13s | 268689 | 1 | 178s | 1419901 |
| | 1e-15 | 1 | 3s | 72524 | 1 | 37s | 348056 |
| | 1e-10 | 1 | $< 1s$ | 11161 | 1 | 3s | 43343 |
| 2 | 1e-20 | 5.9530e-5 | 106s | 1514580 | 3.8544e-9 | 1625s | 9163634 |
| | 1e-15 | 3.3907e-1 | 34s | 479171 | 1.8114e-4 | 729s | 3943370 |
| | 1e-10 | 1 | 6s | 84349 | 9.5890e-1 | 136s | 684309 |
| 3 | 1e-20 | 3.5570e-3 | 577s | 3450157 | 1.3029e-4 | 1995s | 4757788 |
| | 1e-15 | 3.5541e-3 | 201s | 1138686 | 2.8527e-1 | 452s | 1075486 |
| | 1e-10 | 3.4976e-3 | 30s | 165803 | 9.9755e-1 | 41s | 110010 |
| 4 | 1e-20 | 3.3110e-4 | 99s | 1257158 | 1.6235e-5 | 1451s | 7418540 |
| | 1e-15 | 6.7326e-1 | 29s | 374744 | 5.3342e-4 | 581s | 2833833 |
| | 1e-10 | 1 | 5s | 60968 | 9.9189e-1 | 95s | 458195 |

in our case now an exchange of $\lambda$ and $\mu_8$. Then we consider two ad hoc heuristics based on the reasoning that simply increasing the arrival rate without changing any service rate increases the probability of a high overall population. More precisely, we increase the arrival rate by a factor of two and by a factor of three, respectively. The results obtained by the guided state space exploration are given in Table 4.8. As we can see once more, for sufficiently small truncation threshold $\delta$ our algorithm provides very accurate results with quite low computational efforts.

### 4.3.2 Enzymatic futile cycle

In this section we provide numerical results from comprehensive studies of our guided state space exploration for the enzymatic futile cycle benchmark model described by [SPA05] and considered in the context of weighted stochastic simulation algorithms by [KM08, GRP09, DJRGP11]. In particular, this model is small enough to obtain an (up to numerical errors) exact solution. That is, by setting $\delta = 0$ no truncation error is introduced and if we neglect errors due to the numerical integration method (which is reasonable for this model with our step size chosen as the smallest average sojourn time of all states), then we have a 'quasi-exact' solution. Hence, we are able to evaluate the accuracy of the approxima-

tions obtained by the guided state space exploration in terms of their relative errors.

Table 4.9 – Chemical reactions of the enzymatic futile cycle.

$$\begin{array}{rcl} S_1 + S_2 & \underset{c_2}{\overset{c_1}{\rightleftharpoons}} & S_3 \\ S_3 & \xrightarrow{c_3} & S_1 + S_5 \\ S_4 + S_5 & \underset{c_5}{\overset{c_4}{\rightleftharpoons}} & S_6 \\ S_6 & \xrightarrow{c_6} & S_4 + S_2 \end{array}$$

The chemical reactions of the enzymatic futile are given in Table 4.9 with $c_1 = c_2 = c_4 = c_5 = 1, c_3 = c_6 = 0.1$ and initial state $\mathbf{x}_0 = (1, 50, 0, 1, 50, 0)$. The goal is to estimate the probability that before time $t = 100$ the number of molecules of species $S_5$ drops to $\ell$ for some $L \in \{5, 15, 25\}$.

It remains to choose the guidance functions $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_R$ such that the occurrence of $A$ is more likely than with $\alpha_1, \ldots, \alpha_R$ and the CTMC $(\tilde{\mathbf{X}}(t))_{t \geq 0}$ properly guides us to the rare event. For this purpose we borrow ideas from recent weighted stochastic simulation algorithms (wSSAs) for estimating rare event probabilities based on the well known importance sampling technique for variance reduction [San09a, KM08, GRP09, DJRGP11, RDJGP11]. Actually, with importance sampling the goal is to change the probability measure underlying a stochastic system such that certain target events of interest become more likely to occur in simulations. Hence, it is nearby that choices of $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_R$ that are useful for importance sampling simulations are also useful for our guided state space exploration. As we shall see, our guided state space exploration is far less sensitive against the choice of $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_R$ than importance sampling and weighted stochastic simulation approaches.

In the next section we present experimental results where our choice of the guidance functions is inspired by approaches to state-independent importance sampling as taken in weighted stochastic simulation algorithms (wSSAs) [KM08, GRP09, DJRGP11]. We choose guidance functions

$$\tilde{\alpha}_j(\mathbf{x}) := \gamma_j \alpha_j(\mathbf{x}) \tag{4.5}$$

with positive constants $\gamma_j > 0$. Hence, the parameter biasing consists in assigning a constant factor to each reaction type and multiplying the reaction propensity function by this factor, independent of the specific

state in which the reaction occurs. The factors are collected in the biasing parameter vector $\gamma = (\gamma_1, \ldots, \gamma_R)$. Note that $\gamma_j = 1$ for each reaction $\mathcal{R}_j$ whose propensity function is not changed.

In order to keep the choice of the guidance functions as simple as possible, we do not change all propensity functions $\alpha_1, \ldots, \alpha_R$ but only those for which an increase or decrease obviously increases the probability of the rare event of interest. This can be often seen just by inspection. For instance, if we are interested in a certain species reaching a high or low population level, then we can select the reactions where this species is involved as reactant or product, respectively. In many cases, also 'indirect' impacts of other species on a certain target species can be easily seen.

More specifically, for the probability that before time $t = 100$ the number of molecules of species $S_5$ drops to $\ell$ for some $\ell \in \{5, 15, 25\}$ in the enzymatic futile cycle with initial state $\mathbf{x}_0 = (1, 50, 0, 1, 50, 0)$ we should suppress the creation of $S_5$ molecules. This can be accomplished by decreasing the propensity function of reaction $\mathcal{R}_3$, which creates $S_5$ molecules, and increasing the propensity function of reaction $\mathcal{R}_6$, which by creating $S_4$ molecules encourages the consumption of $S_5$ molecules via reaction $\mathcal{R}_4$. A similar approach has been taken in [KM08, GRP09] by setting the parameter biasing vector to $\gamma = (1, 1, 0.5, 1, 1, 2.0)$.

Table 4.10 – Exact solution results for the enzymatic futile cycle.

| $L$ | Pr | $\lvert Sig\rvert$ |
|---|---|---|
| 25 | 1.7382e-07 | 298 |
| 15 | 6.2866e-13 | 338 |
| 5 | 1.0015e-19 | 378 |

We shall generalize this and study $\gamma = (1, 1, \gamma_3, 1, 1, 1/\gamma_3)$ for various choices of $\gamma_3$ with $0 < \gamma_3 \leq 1$. Hence, in essence the propensity function of reaction $\mathcal{R}_3$ is decreased by the factor $\gamma_j$ and the propensity function of reaction $\mathcal{R}_6$ is increased correspondingly by the factor $1/\gamma_3$, while all other propensity functions remain unchanged. In addition we apply guidance functions proposed by [DJRGP11], where the parameter biasing vector $\gamma$ was obtained via the cross-entropy method [DJRGP11, RK13].

In Table 4.10 we list the probabilities $P(A)$ for three different choices of $L$ as well as the size of the set *Sig* of significant states. For the parameter biasing vector we chose $\gamma = (1, 1, \gamma_3, 1, 1, 1/\gamma_3)$, as explained

Table 4.11 – Guided state space exploration results for the enzymatic futile cycle and parameter biasing vector $\gamma = (1, 1, \gamma_3, 1, 1, 1/\gamma_3)$ with varying $\gamma_3$

| $\gamma_3$ | $\delta$ | L=25 | | L=15 | | L=5 | |
|---|---|---|---|---|---|---|---|
| | | rel.err. | $\lvert Sig\rvert$ | rel.err. | $\lvert Sig\rvert$ | rel.err. | $\lvert Sig\rvert$ |
| 1 | 1e-20 | 1.06e-6 | 263 | 2.77e-5 | 303 | 1 | 330 |
| | 1e-15 | 1.17e-5 | 231 | 1 | 266 | 1 | 266 |
| | 1e-10 | 9.55e-1 | 190 | 1 | 190 | 1 | 190 |
| 0.8 | 1e-20 | 1.06e-6 | 239 | 1.82e-6 | 279 | 1.23e-2 | 319 |
| | 1e-15 | 1.12e-6 | 211 | 2.78e-3 | 251 | 1 | 270 |
| | 1e-10 | 1.53e-2 | 173 | 1 | 191 | 1 | 191 |
| 0.65 | 1e-20 | 1.06e-6 | 220 | 1.80e-6 | 260 | 6.44e-6 | 300 |
| | 1e-15 | 1.06e-6 | 191 | 8.26e-6 | 231 | 3.60e-1 | 269 |
| | 1e-10 | 2.46e-4 | 155 | 6.18e-1 | 193 | 1 | 194 |
| 0.5 | 1e-20 | 1.06e-6 | 198 | 1.80e-6 | 238 | 2.63e-6 | 278 |
| | 1e-15 | 1.06e-6 | 167 | 1.81e-6 | 207 | 6.11e-5 | 247 |
| | 1e-10 | 3.88e-6 | 135 | 1.90e-3 | 175 | 1 | 205 |
| 0.35 | 1e-20 | 1.06e-6 | 167 | 1.80e-6 | 207 | 2.63e-6 | 247 |
| | 1e-15 | 1.06e-6 | 142 | 1.80e-6 | 182 | 2.63e-6 | 222 |
| | 1e-10 | 1.14e-6 | 109 | 3.42e-6 | 149 | 4.93e-4 | 189 |
| 0.2 | 1e-20 | 1.06e-6 | 114 | 1.80e-6 | 154 | 2.63e-6 | 194 |
| | 1e-15 | 1.06e-6 | 86 | 1.80e-6 | 126 | 2.63e-6 | 166 |
| | 1e-10 | 6.51e-5 | 48 | 2.60e-6 | 88 | 2.72e-6 | 128 |

before. In Table 4.11 we list the relative error of the approximated rare event probability and the size of the set *Sig* of significant states for different values of $\gamma_3$, $\delta$ and $L$. Note that a relative error of one corresponds to approximating the rare event probability as zero. This is actually the same as what happens in direct simulation when due to the small probability the rare event is not observed and the probability of the non-observed event is estimated as zero. We also considered the parameter biasing vector $\gamma = (1.000, 1.003, 0.320, 1.003, 0.993, 3.008)$ proposed in [DJRGP11] for $L = 25$. The results are given in Table 4.12. We observe a slightly better approximation compared to the one corresponding to $\gamma_3 = 0.35$ even though the number of significant states was less. For all parameters that we chose, the running time of our algorithm was less than one second.

Table 4.12 – Guided state space exploration results for the enzymatic futile cycle for $L = 25$ and the parameter biasing vector $\gamma = (1.000, 1.003, 0.320, 1.003, 0.993, 3.008)$.

| $\delta$ | rel.err. | $\lvert Sig \rvert$ |
|---|---|---|
| 1e-20 | 5.45e-7 | 162 |
| 1e-15 | 7.38e-7 | 138 |
| 1e-10 | 1.10e-6 | 103 |

# 5 Parameter Estimation for Markov Models of Biochemical Reactions

During the last decade stochastic models of networks of chemical reactions have become very popular. The reason is that the assumption that chemical concentrations change deterministically and continuously in time is not always appropriate for cellular processes. In particular, if certain substances in the cell are present in small concentrations the resulting stochastic effects cannot be adequately described by deterministic models. In that case, discrete-state stochastic models are advantageous because they take into account the discrete random nature of chemical reactions. The theory of stochastic chemical kinetics provides a rigorously justified framework for the description of chemical reactions where the effects of molecular noise are taken into account [Gil77]. It is based on discrete-state Markov processes that explicitly represent the reactions as state-transitions between population vectors. When the molecule numbers are large, the solution of the deterministic description of a reaction network and the mean of the corresponding stochastic model agree up to a small approximation error. If, however, species with small populations are involved, then only a stochastic description can provide probabilities of events of interest such as probabilities of switching between different expression states in gene regulatory networks or the distribution of gene expression products. Moreover, even the mean behavior of the stochastic model can largely deviate from the behavior of the deterministic model [LLBB07]. In such cases the parameters of the stochastic model rather then the parameters of the deterministic model have to be estimated [TXGB07, RAT06, UAL10].

Here, we consider noisy time series measurements of the system state as they are available from wet-lab experiments. Recent experimental

imaging techniques such as high-resolution fluorescence microscopy can measure small molecule counts with measurement errors of less than one molecule [GPZC05]. We assume that the structure of the underlying reaction network is known but the stochastic reaction rate constants of the network are unknown parameters. Then we identify rate constants that maximize the likelihood of the time series data. Maximum likelihood estimators are the most popular estimators since they have desirable mathematical properties. Specifically, they become minimum variance unbiased estimators and are asymptotically normal as the sample size increases.

Our main contribution consists in devising an efficient algorithm for the numerical approximation of the likelihood and its derivatives w.r.t. the stochastic reaction rate constants. Furthermore, we show how similar techniques can be used to estimate the initial molecule numbers of a network as well as parameters related to the measurement error. We also present extensive experimental results that give insights about the identifiability of certain parameters. In particular, we consider a simple gene expression model and the identifiability of reaction rate constants w.r.t. varying observation interval lengths and varying numbers of time series. Moreover, for this system we investigate the identifiability of reaction rate constants if the state of the gene cannot be observed but only the number of mRNA molecules. For a more complex gene regulatory network, we present parameter estimation results where different combinations of proteins are observed. In this way we reason about the sensitivity of the estimation of certain parameters w.r.t. the protein types that are observed.

Previous parameter estimation techniques for stochastic models are based on Monte-Carlo sampling [TXGB07, UAL10] because the discrete state space of the underlying model is typically infinite in several dimensions and a priori a reasonable truncation of the state space is not available. Other approaches are based on Bayesian inference which can be applied both to deterministic and stochastic models [BWK08, Hig77, GG10]. In particular, approximate Bayesian inference can serve as a way to distinguish among a set of competing models [TWS+09]. Moreover, in the context of Bayesian inference linear noise approximations have been used to overcome the problem of large discrete state spaces [KFHR09].

Our method is not based on sampling but directly calculates the likelihood

using a dynamic truncation of the state space. More precisely, we first show that the computation of the likelihood is equivalent to the evaluation of a product of vectors and matrices. This product includes the transition probability matrix of the associated continuous-time Markov process, i.e., the solution of the Kolmogorov differential equations (KDEs), which can be seen as a matrix-version of the chemical master equation. Solving the KDEs is infeasible because of the state space of the underlying Markov model is very large or even infinite. Therefore we use an iterative approximation algorithm from Chapter 2 during which the state space is truncated in an on-the-fly fashion, that is, during a certain time interval we consider only those states that significantly contribute to the likelihood.

## 5.1 Parameter inference

Following the notation in [RAT06], we assume that observations of the reaction network are made at time instances $t_1, \ldots, t_L \in \mathbb{R}_{\geq 0}$ where $t_1 < \ldots < t_L$. Since it is unrealistic to assume that all species can be observed, we assume w.l.o.g. that the species are ordered such that we have observations of $X_1, \ldots, X_d$ for some fixed $d$ with $1 \leq d \leq N$, i.e. $O_i(t_\ell)$ is the observed number of species $i$ at time $t_\ell$ for $i \in \{1, \ldots, d\}$ and $\ell \in \{1, \ldots, L\}$. Let $\mathbf{O}(t_\ell) = (O_1(t_\ell), \ldots, O_d(t_\ell))$ be the corresponding vector of observations. Since these observations are typically subject to measurement errors, we assume that $O_i(t_\ell) = X_i(t_\ell) + \xi_i(t_\ell)$ where the error term vectors $\xi(t_\ell)$ are independent and identically normally distributed with mean $\mathbf{0}$ and covariance matrix $\mathbf{\Sigma} = (Cov(\xi_{i_1}, \xi_{i_2}))$, $i_1, i_2 \in \{1, \ldots, N\}$. Note that $X_i(t_\ell)$ is the true population of the $i$-th species at time $t_\ell$. Clearly, this implies that, conditional on $X_i(t_\ell)$, the random variable $O_i(t_\ell)$ is independent of all other observations as well as independent of the history of $\mathbf{X}$ before time $t_\ell$.

We assume further that we do not know the values of the rate constants $\mathbf{c} = (c_1, \ldots, c_R)$ and our aim is to estimate these constants. Similarly, the initial populations $\mathbf{x}(0)$ and the covariance matrix $\mathbf{\Sigma} = (\Sigma_{i_1,i_2})$ of the error terms are unknown and must be estimated.

Let $f$ denote the joint density of $\mathbf{O}(t_1), \ldots, \mathbf{O}(t_L)$ and, by convenient abuse of notation, for a vector $\mathbf{x}_\ell = (x_1, \ldots, x_d)$ let $\mathbf{X}(t_\ell) = \mathbf{x}_\ell$ represent the event that $X_i(t_\ell) = x_i$ for $1 \leq i \leq d$. In other words, $\mathbf{x}(t_\ell) = \mathbf{x}_\ell$ means that the populations of the observed species at time $t_\ell$ equal the

populations of vector $\mathbf{x}_\ell$. Note that this event corresponds to a set of states of the Markov process since $d$ may be smaller than $N$. More precisely, $Pr(\mathbf{X}(t_\ell) = \mathbf{x}_\ell) = \sum_{\tilde{\mathbf{x}}:\tilde{x_i}=x_i, i\leq d} p(\tilde{\mathbf{x}}, t_\ell)$. Now the likelihood of the observation sequence $\mathbf{O}(t_1), \ldots, \mathbf{O}(t_L)$ is given by

$$\begin{aligned} \mathcal{L} &= f\left(\mathbf{O}(t_1), \ldots, \mathbf{O}(t_L)\right) \\ &= \sum_{\mathbf{x}_1} \ldots \sum_{\mathbf{x}_L} f\left(\mathbf{O}(t_1), \ldots, \mathbf{O}(t_L) \mid \mathbf{X}(t_1) = \mathbf{x}_1, \ldots, \mathbf{X}(t_L) = \mathbf{x}_L\right) \\ &\quad Pr(\mathbf{X}(t_1) = \mathbf{x}_1, \ldots, \mathbf{X}(t_L) = \mathbf{x}_L)\,. \end{aligned} \tag{5.1}$$

Note that $\mathcal{L}$ depends on the chosen rate parameters $\mathbf{c}$ and the initial populations $\mathbf{x}(0)$ since the probability measure $Pr(\cdot)$ does. Furthermore, $\mathcal{L}$ depends on $\mathbf{\Sigma}$ since the density $f$ does. When necessary, we will make this dependence explicit by writing $\mathcal{L}(\mathbf{x}(0), \mathbf{c}, \mathbf{\Sigma})$ instead of $\mathcal{L}$. We now seek constants $\mathbf{c}^*$, initial populations $\mathbf{x}(0)^*$ and a covariance matrix $\mathbf{\Sigma}^*$ such that

$$\mathcal{L}(\mathbf{x}(0)^*, \mathbf{c}^*, \mathbf{\Sigma}^*) = \max_{\mathbf{x}(0), \mathbf{c}, \mathbf{\Sigma}} \mathcal{L}(\mathbf{x}(0), \mathbf{c}, \mathbf{\Sigma})$$

where the maximum is taken over vectors $\mathbf{x}(0)$, $\mathbf{c}$ with strictly positive components and all feasible values of $\Sigma_{i_1,i_2}$. This optimization problem is known as the maximum likelihood problem [Lju99]. Note that $\mathbf{x}(0)^*$, $\mathbf{c}^*$ and $\mathbf{\Sigma}^*$ are random variables because they depend on the (random) observations $\mathbf{O}(t_1), \ldots, \mathbf{O}(t_L)$.

If more than one sequence of observations is made, then the corresponding likelihood is the product of the likelihoods of all individual sequences. More precisely, if $\mathbf{O}^k(t_l)$ is the $k$-th observation that has been observed at time instant $t_l$ where $k \in \{1, \ldots, K\}$, then we define $\mathcal{L}_k(\mathbf{x}(0), \mathbf{c}, \mathbf{\Sigma})$ as the probability to observe $\mathbf{O}^k(t_1), \ldots, \mathbf{O}^k(t_L)$ and maximize

$$\prod_{k=1}^{K} \mathcal{L}_k(\mathbf{x}(0), \mathbf{c}, \mathbf{\Sigma}). \tag{5.2}$$

In what follows, we concentrate on expressions for $\mathcal{L}_k(\mathbf{x}(0), \mathbf{c}, \mathbf{\Sigma})$ and its derivatives with respect to $\mathbf{x}(0)$, $\mathbf{c}$ and $\mathbf{\Sigma}$. We first assume $K = 1$ and drop index $k$. We consider the case $K > 1$ later. In (5.1) we sum over all population vectors $x_1, \ldots, x_L$ of dimension $d$ such that $Pr(\mathbf{X}(t_\ell) = \mathbf{x}_\ell, 1 \leq \ell \leq L) > 0$. Since $\mathbf{X}$ has a large or even infinite state space, it is computationally infeasible to explore all possible sequences.

In Section 5.2 we propose an algorithm to approximate the likelihoods and their derivatives by dynamically truncating the state space and using the fact that (5.1) can be written as a product of vectors and matrices. Let $\phi_{\mathbf{\Sigma}}$ be the density of the normal distribution with mean zero and covariance matrix $\mathbf{\Sigma}$. Then

$$\begin{aligned} &f\left(\mathbf{O}(t_1), \ldots, \mathbf{O}(t_L) \mid \mathbf{X}(t_1) = \mathbf{x}_1, \ldots, \mathbf{X}(t_L) = \mathbf{x}_L\right) \\ &= \textstyle\prod_{\ell=1}^{L} f\left(\mathbf{O}(t_\ell) \mid \mathbf{X}(t_\ell) = x_\ell\right) = \prod_{\ell=1}^{L} \phi_{\mathbf{\Sigma}}(\mathbf{O}(t_\ell) - x_\ell). \end{aligned}$$

If we write $w(\mathbf{x}_\ell)$ for $\phi_{\mathbf{\Sigma}}(\mathbf{O}(t_\ell) - \mathbf{x}_\ell)$, then the sequence $\mathbf{x}_1, \ldots, \mathbf{x}_L$ has "weight" $\prod_{\ell=1}^{L} w(\mathbf{x}_\ell)$ and, thus,

$$\mathcal{L} = \sum_{\mathbf{x}_1} \ldots \sum_{\mathbf{x}_L} Pr(\mathbf{X}(t_1) = \mathbf{x}_1, \ldots, \mathbf{X}(t_L) = \mathbf{x}_L) \prod_{\ell=1}^{L} w(\mathbf{x}_\ell). \tag{5.3}$$

Moreover, for the probability of the sequence $\mathbf{x}_1, \ldots, \mathbf{x}_L$ we have

$$Pr(\mathbf{X}(t_1) = \mathbf{x}_1, \ldots, \mathbf{X}(t_L) = \mathbf{x}_L) = p(\mathbf{x}_1, t_1) P_2(\mathbf{x}_1, \mathbf{x}_2) \cdots P_L(\mathbf{x}_{L-1}, \mathbf{x}_L)$$

where $P_\ell(\mathbf{x}, \tilde{\mathbf{x}}) = Pr(\mathbf{X}(t_\ell) = \tilde{\mathbf{x}} \mid \mathbf{X}(t_{\ell-1}) = \mathbf{x})$ for $d$-dimensional population vectors $\mathbf{x}$ and $\tilde{\mathbf{x}}$. Hence, (5.3) can be written as

$$\mathcal{L} = \sum_{\mathbf{x}_1} p(\mathbf{x}_1, t_1) w(\mathbf{x}_1) \sum_{\mathbf{x}_2} P_2(\mathbf{x}_1, \mathbf{x}_2) w(\mathbf{x}_2) \ldots \sum_{\mathbf{x}_L} P_L(\mathbf{x}_{L-1}, \mathbf{x}_L) w(\mathbf{x}_L). \tag{5.4}$$

Assume that $d = N$ and let $P_\ell$ be the matrix with entries $P_\ell(\mathbf{x}, \tilde{\mathbf{x}})$ for all possible states $\mathbf{x}, \tilde{\mathbf{x}}$. Note that $P_\ell$ is the transition probability matrix of $\mathbf{x}$ for time step $t_\ell - t_{\ell-1}$ and thus the general solution $e^{Q(t_\ell - t_{\ell-1})}$ of the Kolmogorov forward and backward differential equations

$$\tfrac{d}{dt} P_\ell = Q P_\ell, \qquad \tfrac{d}{dt} P_\ell = P_\ell Q.$$

In this case, using $\mathbf{p}(t_1) = \mathbf{p}(t_0) P_1$ with $t_0 = 0$, we can write (5.4) in matrix-vector form as

$$\mathcal{L} = \mathbf{p}(t_0) P_1 W_1 P_2 W_2 \cdots P_L W_L \mathbf{e}. \tag{5.5}$$

Here, $\mathbf{e}$ is the vector with all entries equal to one and $W_\ell$ is a diagonal matrix whose diagonal entries are all equal to $w(\mathbf{x}_\ell)$ with $\ell \in \{1, \ldots, L\}$, where $W_\ell$ is of the same size as $P_\ell$.

If $d < N$, then we still have the same matrix-vector product as in (5.5), but define the weight $w(\mathbf{x})$ of an $N$-dimensional population vector as

$$w(x_1, \ldots, x_N) = w(x_1, \ldots, x_d) = \phi_{\mathbf{\Sigma}}(\mathbf{O}(t_\ell) - \mathbf{x}),$$

i.e. the populations of the unobserved species have no influence on the weight.

Since it is in general not possible to analytically obtain parameters that maximize $\mathcal{L}$, we use numerical optimization techniques to find $\mathbf{c}^*$, $\mathbf{x}(0)^*$ and $\mathbf{\Sigma}^*$. Typically, such techniques iterate over values of $\mathbf{c}$, $\mathbf{x}(0)$ and $\mathbf{\Sigma}$, and increase the likelihood $\mathcal{L}(\mathbf{c}, \sigma, \mathbf{\Sigma})$ by following the gradient. Therefore, we need to calculate the derivatives $\frac{\partial}{\partial c_j}\mathcal{L}$, $\frac{\partial}{\partial x_i(0)}\mathcal{L}$ and $\frac{\partial}{\partial \Sigma_{ß_1,ß_2}}\mathcal{L}$. For $\frac{\partial}{\partial c_j}\mathcal{L}$ we obtain

$$\begin{aligned} \tfrac{\partial}{\partial c_j}\mathcal{L} &= \tfrac{\partial}{\partial c_j}\left(\mathbf{p}(t_0)P_1W_1P_2W_2\cdots P_LW_L\mathbf{e}\right) \\ &= \mathbf{p}(t_0)\left(\textstyle\sum_{\ell=1}^{L}\left(\tfrac{\partial}{\partial c_j}P_\ell\right)W_\ell\prod_{\ell'\neq\ell}P_{\ell'}W_{\ell'}\right)\mathbf{e}. \end{aligned} \tag{5.6}$$

The derivatives of $\mathcal{L}$ w.r.t. $x_i(0)$ and $\Sigma_{ß_1,ß_2}$ are derived analogously. The only difference is that $\mathbf{p}(t_0)$ is dependent on $x_i(0)$ and $P_1, \ldots, P_L$ are independent of $\Sigma_{ß_1,ß_2}$ but $W_1, \ldots, W_L$ depend on $\Sigma_{ß_1,ß_2}$. It is also important to note that expressions for partial derivatives of second order can be derived in a similar way. These derivatives can then be used for an efficient gradient-based local optimization.

For $K > 1$ observation sequences we maximize the log-likelihood

$$\log\textstyle\prod_{k=1}^{K}\mathcal{L}_k = \sum_{k=1}^{K}\log\mathcal{L}_k, \tag{5.7}$$

instead of the likelihood in (5.2). Note that the derivatives are then given by

$$\tfrac{\partial}{\partial\lambda}\textstyle\sum_{k=1}^{K}\log\mathcal{L}_k = \sum_{k=1}^{K}\frac{\frac{\partial}{\partial\lambda}\mathcal{L}_k}{\mathcal{L}_k}, \tag{5.8}$$

where $\lambda$ is $c_j$, $x_i(0)$ or $\Sigma_{ß_1,ß_2}$. It is also important to note that only the weights $w(\mathbf{x}_\ell)$ depend on $k$, that is, on the observed sequence $\mathbf{O}^k(t_1), \ldots, \mathbf{O}^k(t_L)$. Thus, when we compute $\mathcal{L}_k$ based on (5.5) we use for all $k$ the same transition matrices $P_1, \ldots, P_L$ and the same initial conditions $\mathbf{p}(t_0)$, but possibly different matrices $W_1, \ldots, W_L$.

## 5.2 Numerical approximation algorithm

In this section, we focus on the numerical approximation of the likelihood and the corresponding derivatives w.r.t. the rate constants $c_1, \ldots, c_R$, the initial populations $\mathbf{x}(0)$ and the covariance matrix $\mathbf{\Sigma} = (\Sigma_{i_1,i_2})$ of the error terms. For sake of simplicity, we assume that there is no correlation between measurement errors of different molecular species, e.g. $\mathbf{\Sigma} = I\sigma^2$ for some $\sigma > 0$. We propose two approximation algorithms for the likelihood and its derivatives, a state-based likelihood approximation (SLA) and a path-based likelihood approximation (PLA). Both are based on a dynamic truncation of the state space as suggested in Chapter 2. They differ in that the PLA method exploits equidistant time series, that is, it is particularly efficient if $h = t_{\ell+1} - t_\ell$ for all $\ell$ and if $\sigma$ is not too large. The SLA algorithm works for arbitrarily spaced time series and is efficient even if $\sigma$ is large.

### 5.2.1 Computation of derivatives

We are interested in the partial derivatives of $\mathbf{p}(t)$ w.r.t. a certain parameter $\lambda$ such as reaction rate constants $c_j$, $j \in \{1, \ldots, R\}$ or initial populations $x_i(0)$, $i \in \{1, \ldots, N\}$. Later, they will be used to maximize the likelihood of observations and to find optimal parameters. In order to explicitly indicate the dependence of $\mathbf{p}(t)$ on $\lambda$ we may write $\mathbf{p}_\lambda(t)$ instead of $\mathbf{p}(t)$ and $p_\lambda(\mathbf{x}, t)$ instead of $p(\mathbf{x}, t)$. We define the row vector $\mathbf{s}_\lambda(t)$ as the derivative of $\mathbf{p}_\lambda(t)$ w.r.t. $\lambda$, i.e.,

$$\mathbf{s}_\lambda(t) = \tfrac{\partial \mathbf{p}_\lambda(t)}{\partial \lambda} = \lim_{\Delta \to 0} \tfrac{\mathbf{p}_{\lambda+\Delta}(t) - \mathbf{p}_\lambda(t)}{\Delta}.$$

We denote the entry in $\mathbf{s}_\lambda(t)$ that corresponds to state $\mathbf{x}$ by $s_\lambda(\mathbf{x}, t)$. Note that we use bold face for vectors. By (1.6), we find that $\mathbf{s}_\lambda(t)$ is the solution of the system of ODEs

$$\tfrac{d}{dt}\mathbf{s}_\lambda(t) = \mathbf{s}_\lambda(t)Q + \mathbf{p}_\lambda(t)\tfrac{\partial}{\partial \lambda}Q, \tag{5.9}$$

when choosing $\lambda = c_j$ for $j \in \{1, \ldots, R\}$. In this case, the initial condition is $s_\lambda(\mathbf{x}, 0) = 0$ for all $\mathbf{x}$ since $p(\mathbf{x}, 0)$ is independent of $c_j$. If the unknown parameter is the $i$-th initial population, i.e., $\lambda = x_i(0)$, then we get

$$\tfrac{d}{dt}\mathbf{s}_\lambda(t) = \mathbf{s}_\lambda(t)Q, \tag{5.10}$$

with initial condition $\mathbf{s}_\lambda(0) = \frac{\partial}{\partial\lambda}\mathbf{p}_\lambda(0)$ since $Q$ is independent of $x_i(0)$. Similar ODEs can be derived for higher order derivatives of the CME.

### 5.2.2 State-based likelihood approximation

The SLA algorithm calculates an approximation of the likelihood based on (5.5) by traversing the matrix-vector product from the left to the right. The main idea behind the algorithm is that instead of explicitly computing the matrices $P_\ell$, we express the vector-matrix product $\mathbf{u}(t_{\ell-1})P_\ell$ as a system of ODEs similar to the CME (cf. Eq. (1.5)). Here, $\mathbf{u}(t_0), \ldots, \mathbf{u}(t_L)$ are row vectors obtained during the iteration over time points $t_0, \ldots, t_L$, that is, we define $\mathcal{L}$ recursively as $\mathcal{L} = \mathbf{u}(t_L)\mathbf{e}$ with $\mathbf{u}(t_0) = \mathbf{p}(t_0)$ and

$$\mathbf{u}(t_\ell) \;=\; \mathbf{u}(t_{\ell-1})P_\ell W_\ell \qquad \text{for all } 1 \le \ell \le L,$$

where $t_0 = 0$. Instead of computing $P_\ell$ explicitly, we solve $L$ systems of ODEs

$$\tfrac{d}{dt}\tilde{\mathbf{u}}(t) = \tilde{\mathbf{u}}(t)Q \tag{5.11}$$

with initial condition $\tilde{\mathbf{u}}(t_{\ell-1}) = \mathbf{u}(t_{\ell-1})$ for the time interval $[t_{\ell-1}, t_\ell)$ where $\ell \in \{1, \ldots, L\}$. After solving the $\ell$-th system of ODEs we set $\mathbf{u}(t_\ell) = \tilde{\mathbf{u}}(t_\ell)W_\ell$ and finally compute $\mathcal{L} = \mathbf{u}(t_L)\mathbf{e}$. Since this is the same as solving the CME for different initial conditions, we can use the dynamic truncation of the state space proposed in Chapter 2. Since the vectors $\tilde{\mathbf{u}}(t_\ell)$ do not sum up to one, we scale all entries by multiplication with $1/(\tilde{\mathbf{u}}(t_\ell)\mathbf{e})$. This simplifies the truncation of the state space using the significance threshold $\delta$ since after scaling it can be interpreted as a probability. In order to obtain the correct (unscaled) likelihood, we compute $\mathcal{L}$ as $\mathcal{L} = \prod_{\ell=1}^{L}(\tilde{\mathbf{u}}(t_\ell)\mathbf{e})$. We handle the derivatives of $\mathcal{L}$ in a similar way. To shorten our presentation, we only consider the derivative $\frac{\partial}{\partial c_j}\mathcal{L}$ in the sequel. An iterative scheme for $\frac{\partial}{\partial\sigma}\mathcal{L}$ is derived analogously. From (5.6) we obtain $\frac{\partial}{\partial c_j}\mathcal{L} = \mathbf{u}_j(t_L)\mathbf{e}$ with $\mathbf{u}_j(t_0) = \mathbf{0}$ and

$$\mathbf{u}_j(t_\ell) \;=\; (\mathbf{u}_j(t_{\ell-1})P_\ell + \mathbf{u}(t_{\ell-1})\tfrac{\partial}{\partial c_j}P_\ell)W_\ell \qquad \text{for all } 1 \le \ell \le L,$$

where $\mathbf{0}$ is the vector with all entries zero. Thus, during the solution of the $\ell$-th ODE in (5.11) we simultaneously solve

$$\tfrac{d}{dt}\tilde{\mathbf{u}}_j(t) = \tilde{\mathbf{u}}_j(t)Q + \tilde{\mathbf{u}}(t)\tfrac{\partial}{\partial c_j}Q \tag{5.12}$$

with initial condition $\tilde{\mathbf{u}}_j(t_{\ell-1}) = \mathbf{u}_j(t_{\ell-1})$ for the time interval $[t_{\ell-1}, t_\ell)$. As above, we set $\mathbf{u}_j(t_\ell) = \tilde{\mathbf{u}}_j(t_\ell)W_\ell$ and obtain $\frac{\partial}{\partial c_j}\mathcal{L}$ as $\mathbf{u}_j(t_L)\mathbf{e}$.

Solving (5.11) and (5.12) simultaneously is equivalent to the computation of the partial derivatives in (5.9) with different initial conditions. Thus, we can use the approximation algorithm proposed in Chapter 2 to approximate $\mathbf{u}_j(t_\ell)$. Our experimental results show that the approximation errors of the likelihood and its derivatives are of the same order of magnitude as those of the transient probabilities and their derivatives (not shown). Note, however, that, if $\sigma$ is small only few states contribute significantly to the likelihood. In this case, truncation strategies based on sorting of vectors are more efficient without considerable accuracy losses since the main part of the likelihood concentrates on very few entries (namely those that correspond to states that are close to the observed populations).

In the case of $K$ observation sequences we repeat the above algorithm in order to sequentially compute $\mathcal{L}_k$ for $k \in \{1, \ldots, K\}$. We exploit (5.7) and (5.8) to compute the total log-likelihood and its derivatives as a sum of individual terms. Obviously, it is possible to parallelize the SLA algorithm by computing $\mathcal{L}_k$ in parallel for all $k$.

In order to find values for which the likelihood becomes maximal, global optimization techniques can be applied. Those techniques usually use a heuristic for different initial values of the parameters and then follow the gradient to find local optima of the likelihood. In this step the algorithm proposed above is used since it approximates the gradient of the likelihood. The approximated global optimum is then chosen as the minimum/maximum of the local optima, i.e, we determine those values of the parameters that give the largest likelihood. Clearly, this is an approximation and we cannot guarantee that the global optimum was found. Note that this would also be the case if we could compute the exact likelihood. If, however, a good heuristic for the starting points is chosen and the number of starting points is large, then it is likely that the approximation is accurate. Moreover, since we have approximated the second derivative of the log-likelihood, we can compute the entries of the Fisher information matrix and use this to approximate the standard deviation of the estimated parameters, i.e., we consider the square root of the diagonal entries of the inverse of a matrix $H$ which is the Hessian matrix of the negative log-likelihood. Assuming that the second derivative of the log-likelihood is computed exactly, these entries asymptotically

tend to the standard deviations of the estimated parameters.

We remark that the approximation proposed above becomes unfeasible if the reaction network contains species with high molecule numbers since in this case the number of states that have to be considered is very large. A numerical approximation of the likelihood is, as the solution of the CME, only possible if the expected populations of all species remain small (at most a few hundreds) and if the dimension of the process is not too large. Moreover, if many parameters have to be estimated, the search space of the optimization problem may become unfeasibly large. It is however straightforward to parallelize local optimizations starting from different initial point.

### 5.2.3 Path-based likelihood approximation

If $\Delta t = t_\ell - t_{\ell-1}$ for all $\ell$ then the matrices $P_1, \ldots, P_L$ in (5.5) are equal to the $\Delta t$-step transition matrix $\mathbf{T}(\Delta t)$ with entries $Pr(\mathbf{X}(t+\Delta t) = \mathbf{x}_2 \mid \mathbf{X}(t) = \mathbf{x}_1)$. Note that since we consider a time-homogeneous Markov process $\mathbf{x}$, the matrix $\mathbf{T}(\Delta t)$ is independent of $t$. The main idea of the PLA method is to iteratively compute those parts of $\mathbf{T}(\Delta t)$ that correspond to state sequences (paths) $\mathbf{x}_1, \ldots, \mathbf{x}_L$ that contribute significantly to $\mathcal{L}$. The algorithm can be summarized as follows, where we omit the argument $\Delta t$ of $\mathbf{T}$ to improve the readability and refer to the entries of $\mathbf{T}$ as $\mathbf{T}(\mathbf{x}_1, \mathbf{x}_2)$:

1. We compute the transient distribution $\mathbf{p}(t_1)$ and its derivatives (w.r.t. $\mathbf{c}$ and $\sigma$) as outlined in Section 5.2.1 using a significance threshold $\delta$.

2. For each state $\mathbf{x}_1$ with significant probability $p(\mathbf{x}_1, t_1)$ we approximate the rows of $\mathbf{T}$ and $\frac{\partial}{\partial c_j}\mathbf{T}$ that correspond to $\mathbf{x}_1$ based on a transient analysis for $\Delta t$ time units. More precisely, if $\mathbf{e}_{\mathbf{x}_1}$ is the vector with all entries zero except for the entry that corresponds to state $\mathbf{x}_1$ which is one, then we solve (1.5) with initial condition $\mathbf{e}_{\mathbf{x}_1}$ for $\Delta t$ time units in order to approximate $\mathbf{T}(\mathbf{x}_1, \mathbf{x}_2)$ and $\frac{\partial}{\partial c_j}\mathbf{T}(\mathbf{x}_1, \mathbf{x}_2)$ for all $\mathbf{x}_2$. During this transient analysis we again apply the dynamic truncation of the state space proposed in Chapter 2 with threshold $\delta$.

3. We then store for each pair $(\mathbf{x}_1, \mathbf{x}_2)$ the (partial) likelihood $a(\mathbf{x}_1, \mathbf{x}_2)$

and its derivatives:

$$\begin{aligned} a(\mathbf{x}_1, \mathbf{x}_2) &= p(\mathbf{x}_1, t_1) \cdot w(\mathbf{x}_1) \cdot \mathbf{T}(\mathbf{x}_1, \mathbf{x}_2) \cdot w(\mathbf{x}_2) \\ \tfrac{\partial}{\partial c_j} a(\mathbf{x}_1, \mathbf{x}_2) &= \tfrac{\partial}{\partial c_j} p(\mathbf{x}_1, t_1) \cdot w(\mathbf{x}_1) \cdot \mathbf{T}(\mathbf{x}_1, \mathbf{x}_2) \cdot w(\mathbf{x}_2) \\ &\quad + p(\mathbf{x}_1, t_1) \cdot w(\mathbf{x}_1) \cdot \tfrac{\partial}{\partial c_j} \mathbf{T}(\mathbf{x}_1, \mathbf{x}_2) \cdot w(\mathbf{x}_2). \end{aligned}$$

4. We reduce the number of considered pairs by sorting $a(\mathbf{x}_1, \mathbf{x}_2)$ for all pairs $(\mathbf{x}_1, \mathbf{x}_2)$ calculated in the previous step and keep the most probable pairs.

5. Next, we repeat steps 2-4, where in step 2 we start the analysis from all states $\mathbf{x}_2$ that are the last element of a pair kept in the previous step. In step 3 we store triples of states, say, $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ and recursively compute their likelihood and the corresponding derivatives by multiplication with $\mathbf{T}(\mathbf{x}_2, \mathbf{x}_3)$ and $w(\mathbf{x}_3)$, i.e., for the likelihood we compute

$$\begin{aligned} a(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) &= a(\mathbf{x}_1, \mathbf{x}_2) \cdot \mathbf{T}(\mathbf{x}_2, \mathbf{x}_3) \cdot w(\mathbf{x}_3) \\ \tfrac{\partial}{\partial c_j} a(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) &= \tfrac{\partial}{\partial c_j} a(\mathbf{x}_1, \mathbf{x}_2) \cdot \mathbf{T}(\mathbf{x}_2, \mathbf{x}_3) \cdot w(\mathbf{x}_3) \\ &\quad + a(\mathbf{x}_1, \mathbf{x}_2) \cdot \tfrac{\partial}{\partial c_j} \mathbf{T}(\mathbf{x}_2, \mathbf{x}_3) \cdot w(\mathbf{x}_3). \end{aligned}$$

   Note that we may reuse some of the entries of $\mathbf{T}$ since they already have been calculated in a previous step. In step 4 we again reduce the number of triples $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ by sorting them according to their likelihood. We then keep the most probable triples, and so on. Note that in step 4 we cannot use a fixed truncation threshold $\delta$ to reduce the number of state sequences (or paths) since their probabilities may become very small as the sequences become longer.

6. We stop the prolongation of paths $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ when the time instance $t_L = \Delta t \cdot L$ is reached and compute an approximation of $\mathcal{L}$ and its derivatives by summing up the corresponding values of all paths (cf. Eq. (5.3)).

If we have more than one observation sequence, i.e., $K > 1$, then we repeat the procedure to compute $\mathcal{L}_k$ for all $k$ and use (5.7) to calculate the total log-likelihood. Note that the contribution of each path $\mathbf{x}_1, \ldots, \mathbf{x}_L$ to $\mathcal{L}_k$ may be different for each $k$. It is, however, likely that the entries of $\mathbf{T}$ can be reused not only during the computation of each single $\mathcal{L}_k$ but also for different values of $k$. If many entries of $\mathbf{T}$ are reused

during the computation, the algorithm performs fast compared to other approaches. For our experimental results in Section 5.3.1, we keep the ten most probable paths in step 4. Even though this enforces a coarse approximation, the likelihood is approximated very accurately if $\sigma$ is small, since in this case only few paths contribute significantly to $\mathcal{L}_k$. On the other hand, if $\sigma$ is large, then the approximation may become inaccurate depending on the chosen truncation strategy. Another disadvantage of the PLA method is that for non-equidistant time series, the performance is slow since we have to compute (parts of) different transition matrices and, during the computation of $\mathcal{L}_k$, the transition probabilities cannot be reused.

## 5.3 Numerical results

In this section we present numerical results of our parameter estimation algorithm applied to a number of models of biochemical reactions. In subsections 5.3.1, 5.3.2, we present experimental results of the SLA and PLA methods. For equidistant time series, we compare our approach to the approximate maximum likelihood (AML) and the singular value decomposition (SVDL) method described by Reinker et al. [RAT06]. Since an implementation of the AML and SVDL method was not available to us, we chose the same examples and experimental conditions for the time series as Reinker et al. and compared our results to those listed in the results section in [RAT06]. We also consider non-equidistant time series. To the best of our knowledge there exists no direct numerical approach for non-equidistant time series with measurement error that is based on the maximum likelihood method. In subsection 5.3.3 we present results of our parameter estimation algorithm applied to enzyme reaction example where alongside with the estimation of reaction rate constants and the observation error we estimate the initial conditions. In subsection 5.3.4 we provide more complex case studies and run extensive numerical experiments to assess the identifiability of certain parameters. In these experiments we assume that not all molecular populations can be observed and estimate parameters for different observation scenarios, i.e., we assume different numbers of observed cells and different observation interval lengths.

For all models of biochemical reaction networks, we generated time series data using Monte-Carlo simulation where we added white noise

to represent measurement errors, i.e. we added random terms to the populations that follow a normal distribution with mean zero and a standard deviation of $\sigma$. Our algorithm for the approximation of the likelihood is implemented in C++ and linked to MATLAB's optimization toolbox [MAT15] which we use to minimize the negative log-likelihood. The global optimization method (Matlab's GlobalSearch [ULP+07]) uses a scatter-search algorithm to generate a set of trial points (potential starting points) and heuristically decides when to perform a local optimization. We ran our experiments on an Intel Core i7 at 2.8 GHz with 8 GB main memory. As an integration method we used the standard explicit fourth-order Runge-Kutta method with the time step chosen as the smallest average sojourn time of all states in *Sig*, that is,

$$h = \min_{\mathbf{x} \in Sig} 1/\sum_{j=1}^{R} \alpha_j(\mathbf{x}).$$

### 5.3.1 Equidistant time series

We consider models of the simple gene expression and the transcription regulation with the reactions given in Table 5.1 and Table 5.2, respectively.

Table 5.1 – Chemical reactions of the simple gene expression model.

$$\begin{array}{lcl} \mathrm{DNA_{ON}} & \overset{c_1}{\underset{c_2}{\rightleftharpoons}} & \mathrm{DNA_{OFF}} \\ \mathrm{DNA_{ON}} & \xrightarrow{c_3} & \mathrm{DNA_{ON}} + \mathrm{mRNA} \end{array}$$

In the equidistant case, the length of the observation intervals is $\Delta t = t_\ell - t_{\ell-1}$ for all $\ell \in \{1, \ldots, L\}$. In Table 5.3 and 5.5 we list the results given in [RAT06] as well as the results of our methods. Reinker et al. do not evaluate the AML method for larger intervals than $\Delta t = 1$ because the approximation error of the AML method becomes huge in that case. Also, the SVDL method performs poor if $\sigma$ is large since it does not include measurement errors in the likelihood. Therefore, no results for $\sigma > 1.0$ are provided in [RAT06] for SVDL. In the first three columns we list $\Delta t$, the number $L$ of observation points and the true standard deviation $\sigma$ of the error terms. In column "Time" we compare the average running time (in seconds) of one parameter estimation (out of 100) for SLA and PLA, i.e., the average running time of the maximization of the likelihood based on $K = 5$ observation sequences. It is not meaningful to compare the running times with those in [RAT06] since different optimization methods are used and experiments were run on different machines. Finally, we list

estimation results for all four methods (if available). We list the average of 100 estimations and the standard deviation of the estimates.

Table 5.2 – Chemical reactions of the transcription regulation model.

$$
\begin{aligned}
\text{mRNA} &\xrightarrow{c_1} \text{mRNA} + \text{M} \\
\text{M} &\xrightarrow{c_2} \emptyset \\
\text{DNA.D} &\xrightarrow{c_3} \text{mRNA} + \text{DNA.D} \\
\text{mRNA} &\xrightarrow{c_4} \emptyset \\
\text{DNA} + \text{D} &\underset{c_6}{\overset{c_5}{\rightleftharpoons}} \text{DNA.D} \\
2\,\text{M} &\underset{c_8}{\overset{c_7}{\rightleftharpoons}} \text{D}
\end{aligned}
$$

For the simple gene expression (Tables 5.3, 5.4) and $\Delta t = 1.0$, we find that SLA and PLA have a similar accuracy for the estimation of $\sigma$ but are consistently more accurate than AML and SVDL for estimating the rate constants. If $\sigma = 0.1$, then the total absolute error for the estimation of $\mathbf{c}$ is 0.041, 0.073, 0.016, 0.018 for AML, SVDL, SLA, PLA, respectively. For $\sigma = 1.0$ we have total absolute errors of 0.081, 0.053, 0.026, 0.021 for AML, SVDL, SLA, PLA. Finally, for $\sigma = 3.0$, AML has a total error of 0.139 while the error for SLA and PLA is 0.017 and 0.041. For $\Delta t = 10$, the results of the SLA and PLA method are accurate even though only 30 observation points are given. Since PLA gives a much coarser approximation, its running time is always shorter (about three to ten times shorter). If $\sigma$ is large, SLA gives more accurate results than PLA.

In Tables 5.5, 5.6 we compare results of the transcription regulation for $\sigma = 0$. Note that, for this example, Reinker et al. only give results for the SVDL method with $\Delta t \leq 1.0$ and $\sigma = 0$. Here, we compare results for $\Delta t = 1.0$ since in this case the SVDL method performs best compared to smaller values of $\Delta t$. The SLA and PLA method consistently perform better than the SVDL method since they approximate the likelihood more accurately. If $\sigma = 0$, then the accuracy of SLA and PLA is the same (up to the fifth digit). Therefore the results of SLA and PLA are combined in Tables 5.5, 5.6. The running time of SLA is, however, much slower since it does not reuse the entries of the transition probability matrix $\mathbf{T}$. For $\Delta t = 1.0$, one parameter estimation based on $K = 5$ observations takes about 30 minutes for SLA and only about 2.4 minutes for PLA. For $\Delta t = 10.0$ we have running times about 5 hours(SLA) and 27 minutes (PLA). As for the gene expression example, we expect for larger values

of $\sigma$ the results of SLA to be more accurate than those of PLA.

Table 5.3 – Average of parameter estimates for the simple gene expression model using equidistant time series.

| $\Delta t$ ($L$) | $\sigma$ | method | run time | $c_1^*$ | $c_2^*$ | $c_3^*$ | $\sigma^*$ |
|---|---|---|---|---|---|---|---|
| 1.0 (300) | 0.1 | AML | – | 0.0268 | 0.1523 | 0.3741 | 0.1012 |
| | | SVDL | – | 0.0229 | 0.1573 | 0.4594 | – |
| | | SLA | 29.4 | 0.0297 | 0.1777 | 0.3974 | 0.1028 |
| | | PLA | 2.2 | 0.0300 | 0.1629 | 0.3892 | 0.1010 |
| | 1.0 | AML | – | 0.0257 | 0.1409 | 0.3461 | 1.0025 |
| | | SVDL | – | 0.0295 | 0.1321 | 0.3842 | – |
| | | SLA | 8.3 | 0.0278 | 0.1868 | 0.3946 | 0.9976 |
| | | PLA | 1.8 | 0.0278 | 0.1810 | 0.3938 | 0.9938 |
| | 3.0 | AML | – | 0.0250 | 0.1140 | 0.3160 | 3.0292 |
| | | SVDL | – | – | – | – | – |
| | | SLA | 11.1 | 0.0285 | 0.1755 | 0.3938 | 2.9913 |
| | | PLA | 1.7 | 0.0275 | 0.1972 | 0.3894 | 3.0779 |
| 10.0 (30) | 0.1 | AML | – | – | – | – | – |
| | | SVDL | – | – | – | – | – |
| | | SLA | 40.9 | 0.0273 | 0.1788 | 0.3931 | 0.1086 |
| | | PLA | 5.2 | 0.0277 | 0.1782 | 0.4057 | 0.1234 |
| | 1.0 | AML | – | – | – | – | – |
| | | SVDL | – | – | – | – | – |
| | | SLA | 10.2 | 0.0283 | 0.1787 | 0.4018 | 0.9898 |
| | | PLA | 3.5 | 0.0243 | 0.1665 | 0.4031 | 1.0329 |
| | 3.0 | AML | – | – | – | – | – |
| | | SVDL | – | – | – | – | – |
| | | SLA | 12.3 | 0.0300 | 0.1960 | 0.4025 | 2.9402 |
| | | PLA | 4.2 | 0.0210 | 0.1511 | 0.4042 | 3.0629 |

### 5.3.2 Non-equidistant time series

Finally, we consider non-equidistant time series, which can only be handled by the SLA method. During the Monte-Carlo simulation, we generate non-equidistant time series by iteratively choosing $t_{\ell+1} = t_\ell + \mathcal{U}(0, 5)$, where $\mathcal{U}(0, 5)$ is a random number that is uniformly distributed on $(0, 5)$ and $t_0 = 0$. Note that the intervals are not only different within an observation sequence but also for different $k$, i.e., the times $t_1, \ldots, t_L$ depend on the

number $k$ of the corresponding sequence. We consider the transcription regulation model with $\sigma = 1.0$ and $K = 5$ as this is our most complex example. Note that, since the accuracy of the estimation decreases as $\sigma$ increases, we cannot expect a similar accuracy as in Tables 5.5, 5.6. For a time horizon of $T = 500$ the average number of observation points per sequence is $L = 500/2.5 = 200$. The estimates computed by SLA are $c_1^* = 0.0384(0.0343)$, $c_2^* = 0.0010(0.0001)$, $c_3^* = 0.0642(0.0249)$, $c_4^* = 0.0044(0.0047)$, $c_5^* = 0.0273(0.0073)$, $c_6^* = 0.5498(0.1992)$, $c_7^* = 0.0890(0.0154)$, $c_8^* = 0.5586(0.0716)$, and $\sigma^* = 0.9510(0.0211)$, where we averaged over 100 repeated estimations and give the standard deviation in brackets. Recall that the true constants are $c_1 = 0.043$, $c_2 = 0.0007$, $c_3 = 0.0715$, $c_4 = 0.00395$, $c_5 = 0.02$, $c_6 = 0.4791$, $c_7 = 0.083$, and $c_8 = 0.5$. The average running time of one estimation was 19 minutes.

### 5.3.3 Estimation of initial conditions

We consider the enzyme reaction example with the reactions given in Table 1.1 and initial molecular populations $(E(0), S(0), C(0), P(0)) = (20, 10, 0, 0)$ for E, S, C, and P. We chose rate constants $\mathbf{c} = (1, 1, 0.1)$ and a time horizon of $T = 10$. In Fig. 5.1 we plot the expected number of complex molecules and two observation sequences that we simulated using $\sigma^2 = 1$ and $\sigma^2 = 3$. In Tables 5.7 and 5.8 we list the estimation results for different numbers of observation points $L$ yielding different lengths $\Delta t$ of observation intervals. We also vary the measurement error $\sigma^2$. In the third column we list the execution time of a single estimation run.

We list the average and standard deviation of estimated rate constants in columns 4-6. For $\Delta t = 0.1$ and $\sigma^2 = 0.1$ the maximum absolute error of $\mathbf{c}$ is 0.0852, for $\Delta t = 0.1$ and $\sigma^2 = 1$ we have 0.0402, and for $\Delta t = 0.1$ and $\sigma^2 = 3$ it is 0.1910. In case if $\Delta t = 1$ we get a coarser estimation of $c_1$ and $c_2$, but the estimation of $c_3$ is still very accurate. This can be explained by the fact that the third reaction is slower. In the 7th and 8th columns we list the estimated initial molecular counts of type E and S respectively. We assume that we know that initially neither complex molecules nor proteins are present in the system. The intervals for initial molecular counts were chosen as $[0, 50]$. For $\sigma^2 \leq 1$ the maximum absolute error is 0.1295, if $\sigma^2 = 3$ the error is 0.6815. Finally, we list the estimated covariance of the measurement error. For $\Delta t = 0.1$ the maximum absolute

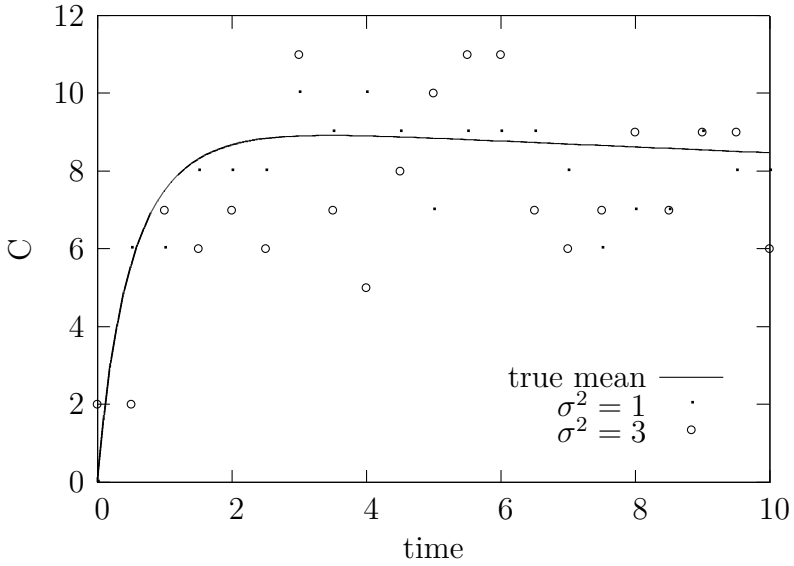



Figure 5.1 – Expected number of complex molecules and two observation sequences for $\sigma^2 = 1$ and $\sigma^2 = 3$.

error is 0.0568 and for $\Delta t = 1$ we have 0.1224.

### 5.3.4 Parameter identifiability

**Simple Gene Expression** We consider the simple gene expression model (Table 5.1) with same parameters chosen as Reinker et al.[RAT06] multiplied by a factor of 10, i.e., $\mathbf{c} = (0.270, 1.667, 4.0)$ and as the initial condition we have 10 mRNA molecules and the DNA is inactive. We generated $K$ observation sequences of length $T = 100.0$ and observed all species at $L$ equidistant observation time points. We added white noise with standard deviation $\sigma = 1.0$ to the observed mRNA molecule numbers at each observation time point. For the case $K = 5$, $L = 100$ we plot the generated observation sequences in Figure 5.2 (a).

We estimated the reaction rate constants, the initial molecule numbers, and the parameter $\sigma$ of the measurement errors for the case $K = 5, L = 100$ where we chose the interval $[10^{-5}, 10^3]$ as a constraint for the rate constants, the interval $[0, 100]$ for the initial number of mRNA molecules and $[0, 5]$ for $\sigma$. Since we use a global optimization method, the running time of our method depends on the number of trial points generated by GlobalSearch. In Figure 5.3 we plot the trial points (red points) and local optimization runs (differently colored lines) for the case of 10 (a),

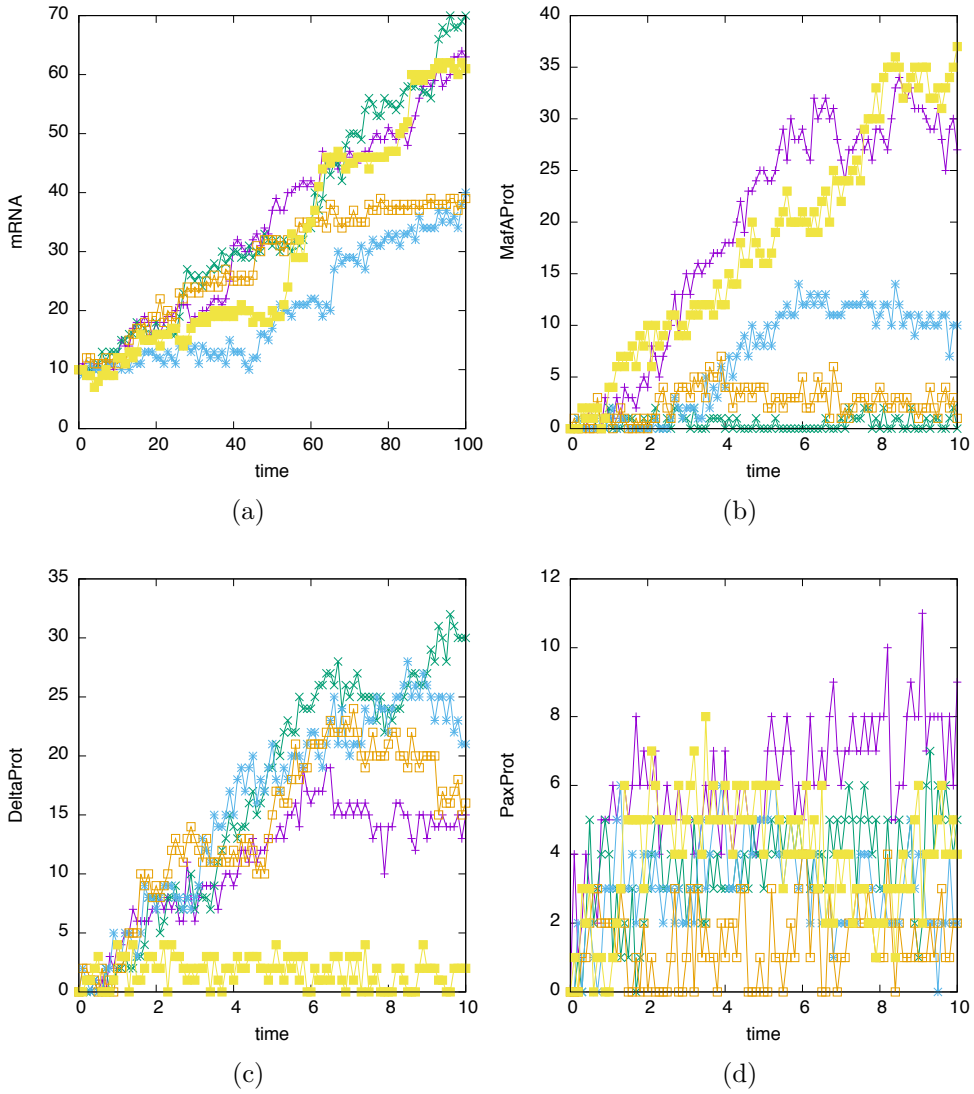



Figure 5.2 – Generated observation sequences for the gene expression (a) and multi-attractor (b)-(d) models. Each plot shows $K = 5$ sequences with $L = 100$ time points.

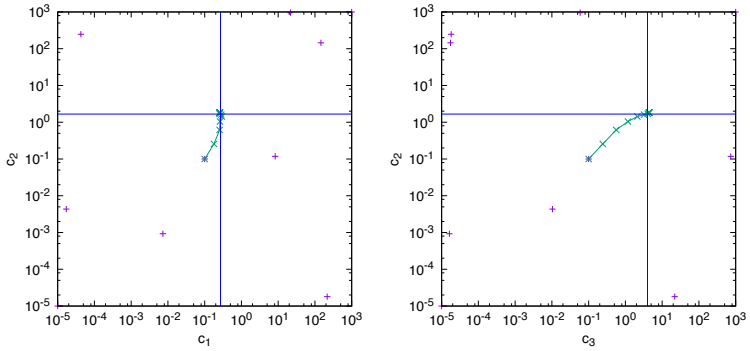
100 (b) and 1000 (c) trial points. The intersection of the dashed blue lines represents the location of the original parameters. In the case of 10 trial points, the running time was about one minute and the local optimization was performed only once. In the case of 100 and 1000 trial points, the running times were about 22 minutes and 1.9 hours respectively and several local optimization runs converged in nearly the same point. However, we remark that in general the landscape of the target function might have multiple local minima and require more trial points resulting

in longer running times.

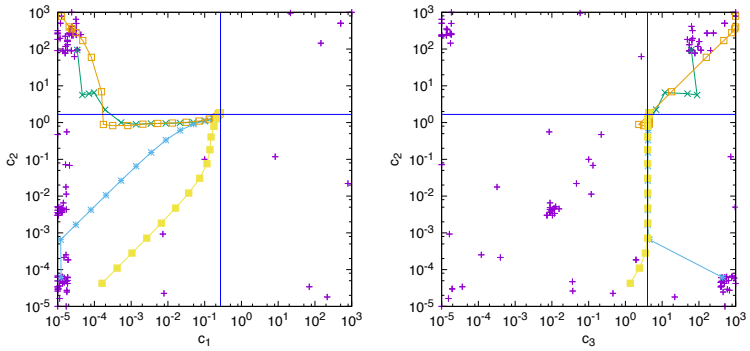
We ran experiments for varying values of $K$ and $L$ ($K, L \in \{1, 2, 5, 10, 20, 50, 100\}$) to get insights whether for this network it is more advantageous to have many observation sequences with long observation intervals or few observation sequences with a short time between two successive observations. In addition, we ran the same experiments with the restriction that only the number of mRNA molecules was observable but not the state of the gene. In both cases we approximated the standard deviations of our estimators as a measure of quality by repeating our estimation procedure 100 times and by the Fisher information matrix as explained at the end of the previous section. We used 100 trial points for the global optimization procedure and chose tighter constraints than above for the rate constants ($[0.01, 1]$ for $c_1$ and $[0.1, 10]$ for $c_2, c_3$) to have a convenient total running time.

The results are depicted in Figure 5.4 for the fully observable system and in Figure 5.5 for the restricted system, where the state of the gene was not visible. In these figures we present the estimations of the parameters $c_1$, $c_2$, $c_3$, $\sigma$, and an estimation of the initial condition, i.e. the number of mRNA molecules at time point $t = 0$. Moreover, we give the total running time of the procedure (Figures 5.4(f) and 5.5(f)). Our results are plotted as a gray landscape for all combinations of $K$ and $L$. The estimates are bounded by a red grid enclosing an environment of one standard deviation around the respective average over all 100 estimates that we approximated. The real value of the parameter is indicated by a dotted blue rectangle.

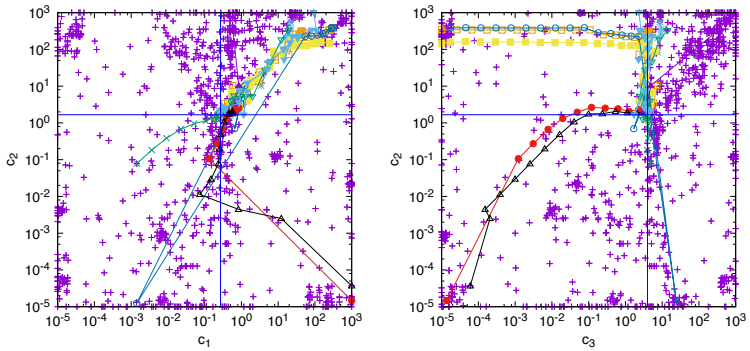
At first, we remark that neither the quality of the estimation nor the running time of our algorithm is significantly dependent on whether we observe the state of the gene in addition to the mRNA level or not. Moreover, concerning the estimation of all of the parameters, one can witness that the estimates converge more quickly against the real values along the $K$ axis than the $L$ axis and also the standard deviations decrease faster. Consequently, at least for the gene expression model, it is more advantageous to increase the number of observation sequences, than the number of measurements per sequence. For example, $K = 100$ sequences with only one observation each already provide enough information to estimate $c_1$ up to a relative error of around 2.1%. Unfortunately, in this case the computation time is the highest since we have to compute $K$




(a)

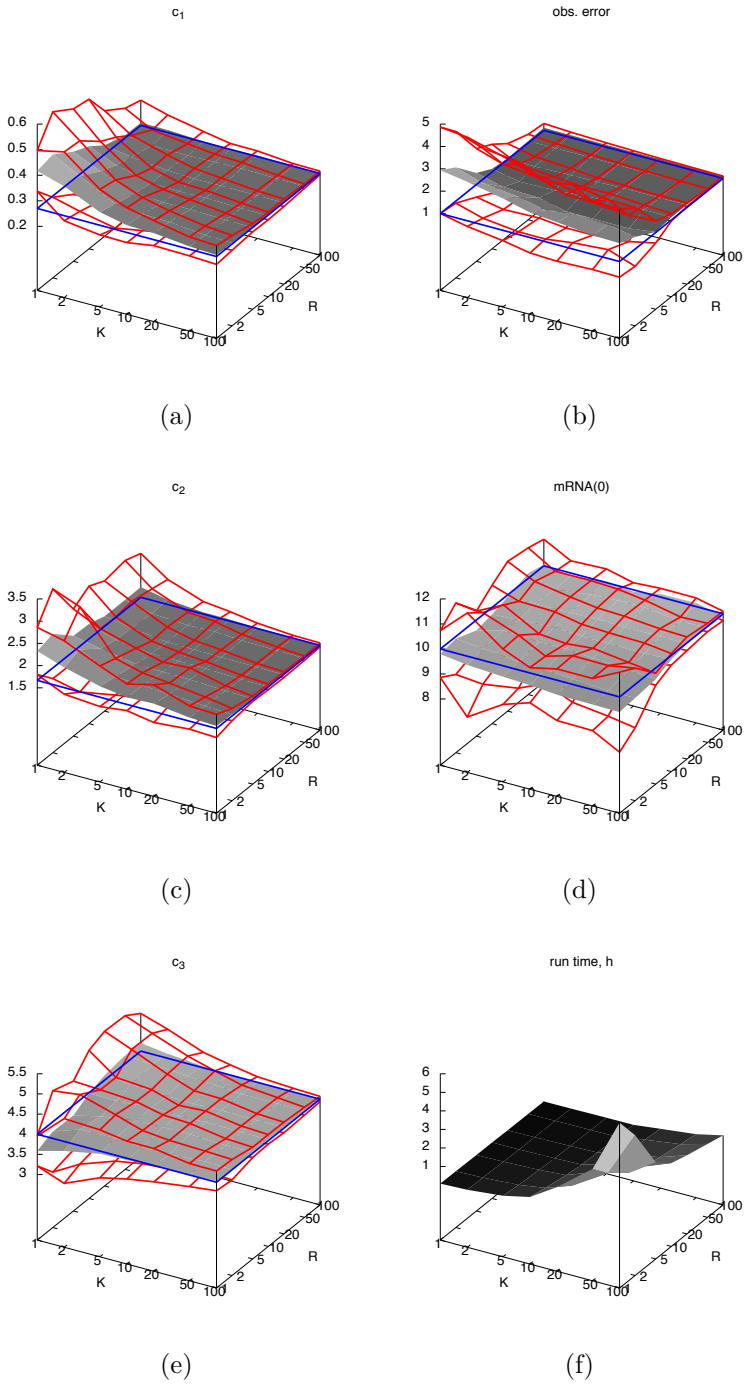



(b)




(c)

Figure 5.3 – Start points and gradient convergence of the optimization procedure for the gene expression example: Red pluses show the potential start points. We use 10, 100, and 1000 start points in case (a), (b), and (c), respectively. The markers that are connected by lines show the iterative steps of the gradient convergence while the dashed blue line shows the true values of the parameters. We chose $K = 5, L = 100$ and assume that the parameters are in the range $[10^{-5}, 10^{3}]$.




(a) (b) (c) (d) (e) (f)

Figure 5.4 – Results of the gene expression case study with observable gene state. The dotted blue rectangle gives the true value of $c_1$, $c_2$, $c_3$, $\sigma$ (obs. error), and mRNA(0). The red grid corresponds to the approximated standard deviation of the estimators.

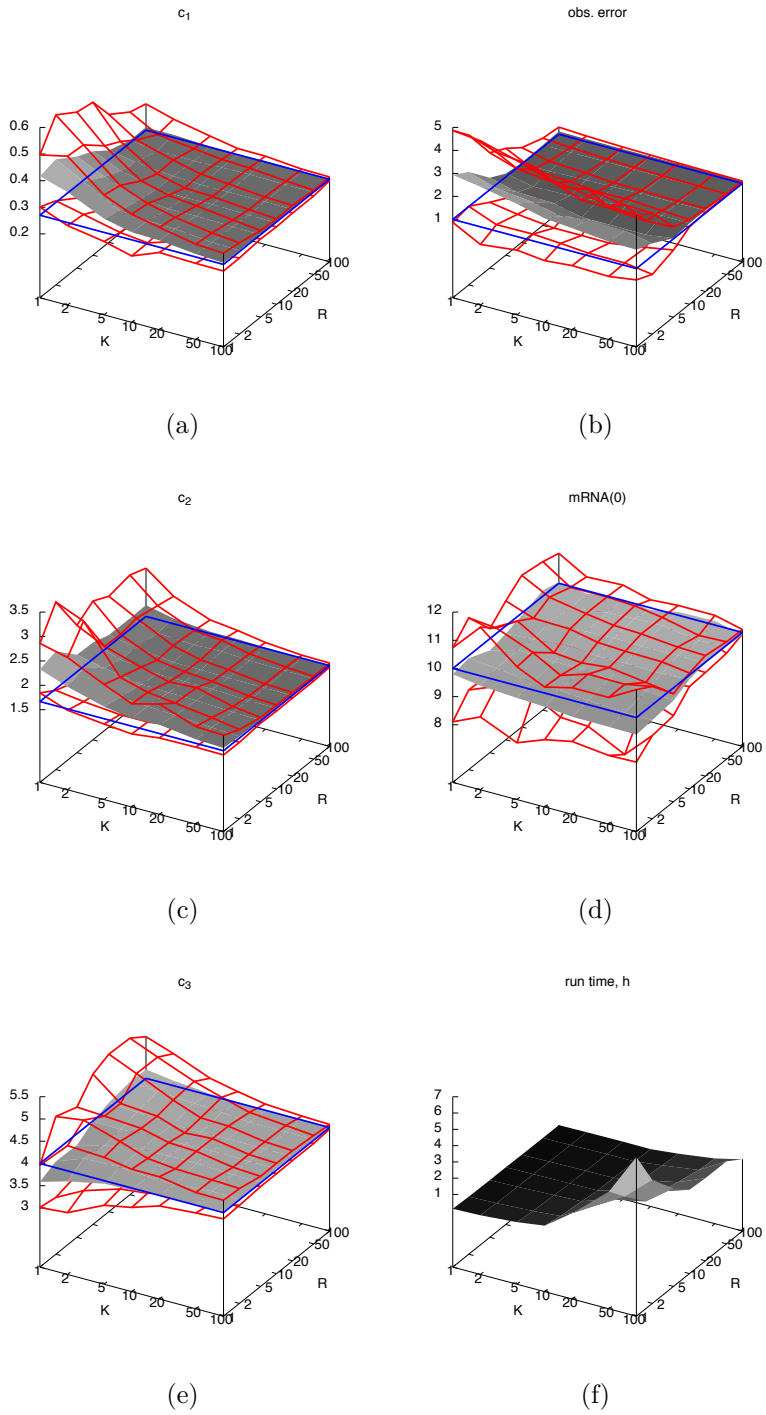



Figure 5.5 – Results of the gene expression case study (as in Figure 5.4) but the state of the gene is not observed.

individual likelihoods (one for each observation sequence). Moreover, if

$L$ is small then the truncation of the state space is less efficient. The reason is that we have to integrate for a long time until we multiply with the weight matrix $W_\ell$. After this multiplication we decide which states contribute significantly to the likelihood and which states are neglected. We can, however, trade off accuracy against running time by varying $K$.

For the measurement noise parameter $\sigma$ we see that it is more advantageous to increase $L$. Even five observation sequences with a high number of observations per sequence ($L = 100$) suffice to estimate the noise up to a relative error of around 10.2%. For the estimation of the initial conditions, both $K$ and $L$ seem to play an equally important role.

The standard deviations of the estimators give information about the accuracy of the estimation. In order to approximate the standard deviation we used statistics over 100 repeated experiments. In a realistic setting one would rather use the Fisher information matrix to approximate the standard deviation of the estimators since it is in most cases difficult to observe $100 \cdot K$ observation sequences of a real system. Therefore we compare the results of one experiment with $K$ observation sequences and standard deviations approximated using the Fisher information matrix to the case where the experiment is repeated 100 times. The results for varying values of $K$ and $L$ are given in Table 5.9. We observe that the approximation using the Fisher information matrix is in most cases close to the approximation based on 100 repetitions as long as $K$ and $L$ are not too small. This comes from the fact that the Fisher information matrix converges to the true standard deviation as the sample size increases.

**Multi-attractor Model** Our final example is a part of the multi-attractor model considered by Zhou et al. [ZBH11]. It consists of the three genes MafA, Pax4, and $\delta$-gene, which interact with each other as illustrated in Figure 5.6. The corresponding proteins bind to specific promoter regions on the DNA and (de-)activate the genes. The reaction network has $2^3$ different gene states, also called modes, since each gene can be on or off. It is infinite in three dimensions since for the proteins there is no fixed upper bound. The edges between the nodes in Figure 5.6 show whether the protein of a specific gene can bind to the promoter region of another gene. Moreover, edges with normal arrow heads correspond to binding without inhibition while the edges with line heads show inhibition.

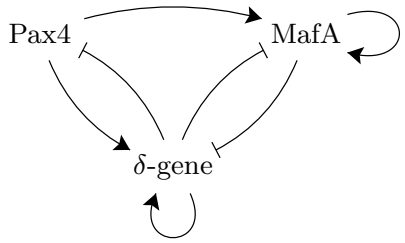



Figure 5.6 – Illustration of the multi-attractor model.

We list all 24 reactions in Table 5.10. For simplicity we first assume that there is a common rate constant for all protein production reactions ($p$), for all protein degradations ($d$), binding ($b$), and unbinding ($u$) reactions. We further assume that initially all genes are active and no proteins are present. For the rate constants we chose $\mathbf{c} = (p, d, b, u) = (5.0, 0.1, 1.0, 1.0)$ and generated $K \in \{1, 5\}$ sample paths of length $T = 10.0$. We added normally distributed noise with zero mean and standard deviation $\sigma = 1.0$ to the protein levels at each of the $L = 100$ observation time points. Plots of the generated observation sequences are presented in Figure 5.2 (b)-(d) for the case $K = 5$. For the global optimization we used 10 trial points. We chose the interval $[0.1, 10]$ as a constraint for the rate constants $p, b, u$ and the interval $[0.01, 1]$ for $d$. We estimated the parameters for all $2^3 - 1 = 7$ possibilities of observing or not observing the three protein numbers where at least one of them had to be observable. In addition we repeated the parameter estimation for the fully observable system where in addition to the three proteins also the state of the genes was observed. The results are depicted in Figure 5.7 where the x-axis of the plots refers to the observed proteins. For instance, the third entry on the x-axis of the plot in Figure 5.7(a) shows the result of the estimation of parameter $c_1 = 5$ based on observation sequences where only the molecule numbers of the proteins MafAProt and DeltaProt were observed. For this case study, we used the Fisher information matrix to approximate the standard deviations of our estimators, plotted as bars in Figure 5.7 with the estimated parameter as midpoint. The fully observable case is labelled by "full".

We observe in Figure 5.7 that as expected the accuracy of the estimation and the running time of our algorithm is best when we have full observability of the system and gets worse with an increasing number of unobservable species. Still the estimation quality is very high when five observation sequences are provided for almost all combinations and

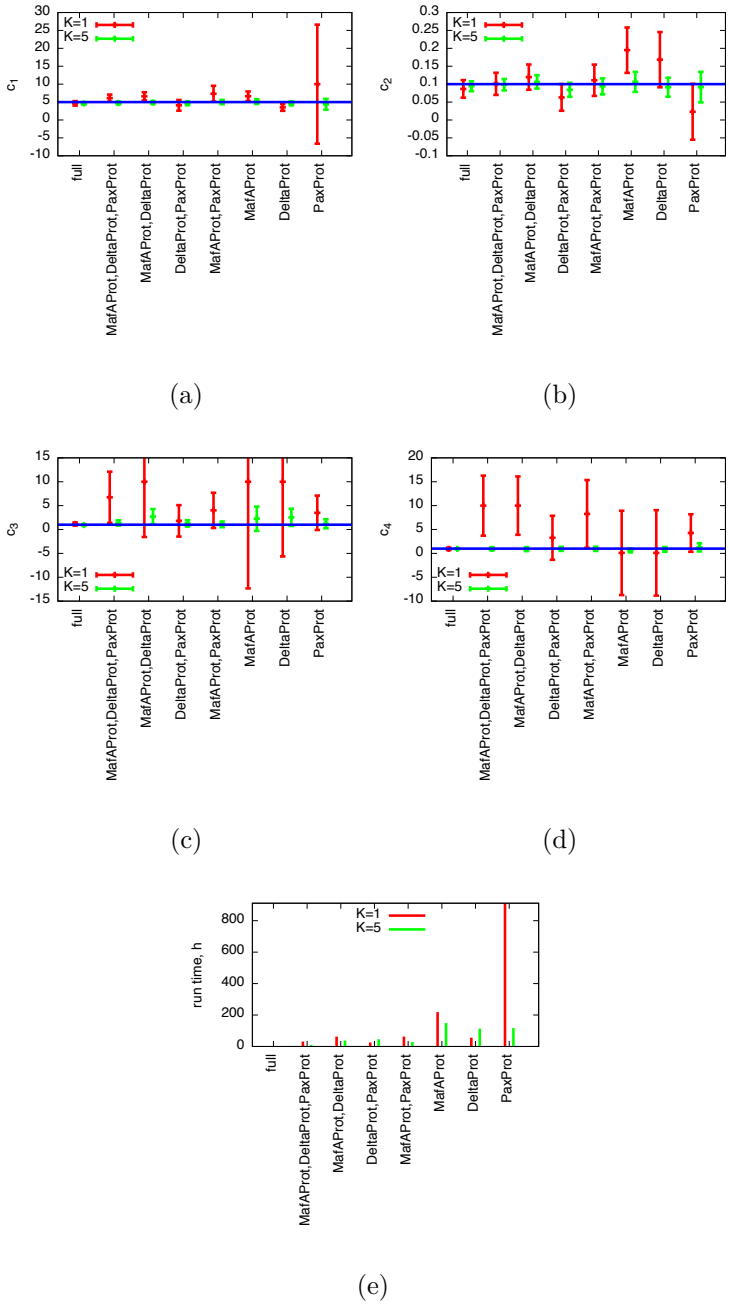
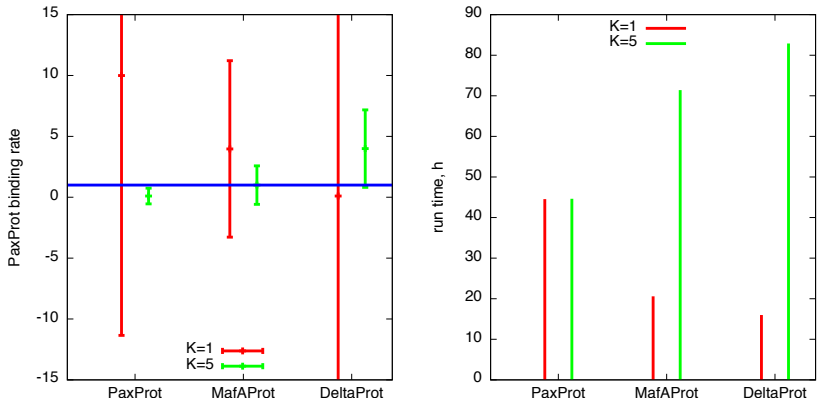
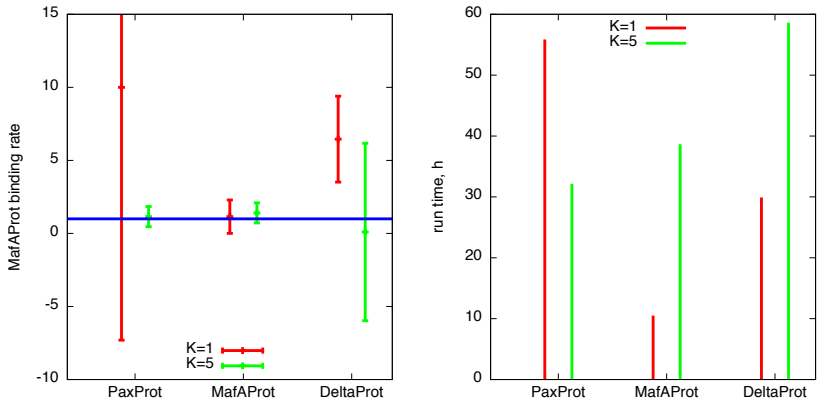


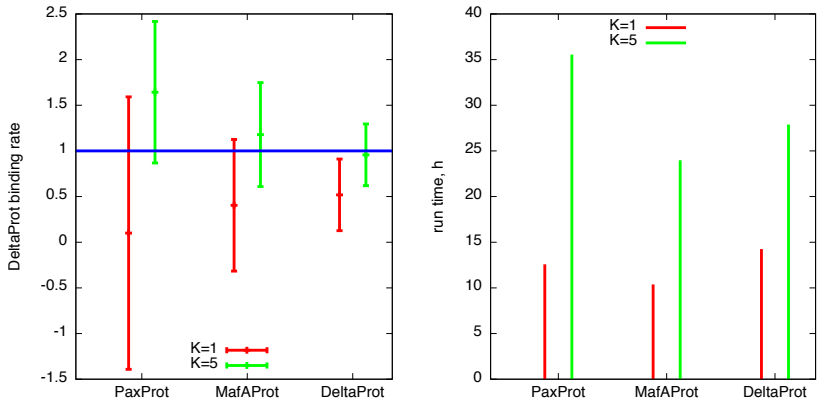

Figure 5.7 – Parameter estimation results for the multi-attractor model. The x-axis shows the species that were observed during the estimation procedure. The dotted blue line corresponds to the true value of $c_1$, $c_2$, $c_3$, and $c_4$, respectively. The error bars in (a)-(d) show the mean (plus/minus the standard deviation) of the estimators. In (e) we plot the running time of the estimation procedure.




(a)




(b)




(c)

Figure 5.8 – Results of the multi-attractor (as in Figure 5.7), but we estimate the binding rate of each protein independently.

parameters. When only one observation sequence is given ($K = 1$), the parameter estimation becomes unreliable and time consuming. This comes from the fact that the quality of the approximation highly depends on the generated observation sequence. It is possible to get much better and faster approximations with a single observation sequence. However, we did not optimize our results but generated one random observation sequence and ran our estimation procedure once based on this.

Recall that we chose common parameters $p$, $d$, $b$, $u$ for production, degradation, and (un-)binding for all three protein species. Next we "decouple" the binding rates and estimate the binding rate of each protein independently. We illustrate our results in Figure 5.8. Again, in case of a single observation sequence ($K = 1$) the estimation is unreliable in most cases. If the true value of the parameter is unknown, then the high standard deviation shows that more information (more observation sequences) is necessary to estimate the parameter. In order to estimate the binding rate of PaxProt, we see that observing MafAProt yields the best result while for the binding rate of MafAProt observing PaxProt is best. Only for the binding rate of DeltaProt, the best results are obtained when the corresponding protein (DeltaProt) is observed. The running times of the estimation procedure are between 10 and 80 hours, usually increase with $K$ and depend on the observation sequences.

In Table 5.11 we list the results of estimating the production rate 5.0 in the multi-attractor model where we chose $L = 100$. More precisely, we estimated the production rate of each protein independently when the other two proteins were observed. Since the population of the PaxProt is significantly smaller than the populations of the other two proteins, its production rate is more difficult to estimate. The production rate of MafAProt is accurately estimated even if only a single observation sequence is considered. For estimating the production rate of DeltaProt, $K = 5$ observation sequences are necessary to get an accurate result.

Finally, we remark that for the multi-attractor model it seems difficult to predict whether for a given parameter the observation of a certain set of proteins yields a good accuracy or not. It can, however, be hypothesized that, if we want to accurately estimate the rate constant of a certain chemical reaction, then we should observe as many of the involved species as possible. Moreover, it is reasonable that constants of reactions that occur less often are more difficult to estimate (such as the production of

PaxProt). In such a case more observation sequences are necessary to provide reliable information about the speed of the reaction.

Table 5.4 – Standard deviation of parameter estimates for the simple gene expression model using equidistant time series.

| $\Delta t$ ($L$) | $\sigma$ | method | $c_1^*$ | $c_2^*$ | $c_3^*$ | $\sigma^*$ |
|---|---|---|---|---|---|---|
| 1.0 (300) | 0.1 | AML | 0.0061 | 0.0424 | 0.0557 | 0.0031 |
| | | SVDL | 0.0041 | 0.0691 | 0.1923 | – |
| | | SLA | 0.0051 | 0.0361 | 0.0502 | 0.0612 |
| | | PLA | 0.0124 | 0.0867 | 0.0972 | 0.0792 |
| | 1.0 | AML | 0.0054 | 0.0402 | 0.0630 | 0.0504 |
| | | SVDL | 0.0102 | 0.0787 | 0.2140 | – |
| | | SLA | 0.0047 | 0.0339 | 0.0419 | 0.0476 |
| | | PLA | 0.0041 | 0.0294 | 0.0315 | 0.0465 |
| | 3.0 | AML | 0.0065 | 0.0337 | 0.0674 | 0.1393 |
| | | SVDL | – | – | – | – |
| | | SLA | 0.0043 | 0.0346 | 0.0508 | 0.0733 |
| | | PLA | 0.0086 | 0.0902 | 0.0722 | 0.0887 |
| 10.0 (30) | 0.1 | AML | – | – | – | – |
| | | SVDL | – | – | – | – |
| | | SLA | 0.0069 | 0.04786 | 0.0599 | 0.0630 |
| | | PLA | 0.0080 | 0.0517 | 0.0678 | 0.0523 |
| | 1.0 | AML | – | – | – | – |
| | | SVDL | – | – | – | – |
| | | SLA | 0.0070 | 0.0523 | 0.0681 | 0.0829 |
| | | PLA | 0.0057 | 0.0400 | 0.0638 | 0.0859 |
| | 3.0 | AML | – | – | – | – |
| | | SVDL | – | – | – | – |
| | | SLA | 0.0110 | 0.0788 | 0.0689 | 0.1304 |
| | | PLA | 0.0054 | 0.0534 | 0.0616 | 0.2249 |

Table 5.5 – Average of parameter estimates for the transcription regulation model using equidistant time series.

| $\Delta t$ $(L)$ | method | $c_1^*$ | $c_2^*$ | $c_3^*$ | $c_4^*$ |
|---|---|---|---|---|---|
| 1.0 (500) | SVDL | 0.0477 | 0.0006 | 0.0645 | 0.0110 |
| | SLA/PLA | 0.0447 | 0.0007 | 0.0677 | 0.0034 |
| 10.0 (50) | SLA/PLA | 0.0417 | 0.0005 | 0.0680 | 0.0038 |

| | | $c_5^*$ | $c_6^*$ | $c_7^*$ | $c_8^*$ |
|---|---|---|---|---|---|
| 1.0 (500) | SVDL | 0.0159 | 0.2646 | 0.0149 | 0.0615 |
| | SLA/PLA | 0.0193 | 0.4592 | 0.0848 | 0.5140 |
| 10.0 (50) | SLA/PLA | 0.0188 | 0.4359 | 0.0836 | 0.4892 |

Table 5.6 – Standard deviation of parameter estimates for the transcription regulation model using equidistant time series.

| $\Delta t$ $(L)$ | method | $c_1^*$ | $c_2^*$ | $c_3^*$ | $c_4^*$ |
|---|---|---|---|---|---|
| 1.0 (500) | SVDL | 0.0155 | 0.0004 | 0.0190 | 0.0195 |
| | SLA/PLA | 0.0036 | 0.0001 | 0.0115 | 0.0014 |
| 10.0 (50) | SLA/PLA | 0.0069 | 0.0002 | 0.0075 | 0.0026 |

| | | $c_5^*$ | $c_6^*$ | $c_7^*$ | $c_8^*$ |
|---|---|---|---|---|---|
| 1.0 (500) | SVDL | 0.0107 | 0.0761 | 0.0143 | 0.0332 |
| | SLA/PLA | 0.0008 | 0.0169 | 0.0024 | 0.0166 |
| 10.0 (50) | SLA/PLA | 0.0039 | 0.0822 | 0.0016 | 0.0164 |

Table 5.7 – Average of parameter estimates for the enzyme reaction network.

| $\Delta t$ $(L)$ | $\sigma^2$ | run time | $c_1^*$ | $c_2^*$ | $c_3^*$ | $E(0)^*$ | $S(0)^*$ | $\sigma^{2,*}$ |
|---|---|---|---|---|---|---|---|---|
| 0.1(100) | 0.1 | 704s | 1.0640 | 1.0852 | 0.0975 | 19.9978 | 10.0020 | 0.1214 |
| | 1.0 | 526s | 1.0577 | 0.9598 | 0.1050 | 19.9927 | 9.9892 | 0.9610 |
| | 3.0 | 464s | 1.1062 | 1.1910 | 0.1058 | 19.9861 | 9.5770 | 3.0568 |
| 1.0(10) | 0.1 | 467s | 0.9198 | 0.8124 | 0.1053 | 19.9909 | 9.9825 | 0.1705 |
| | 1.0 | 553s | 1.0712 | 0.8488 | 0.0914 | 19.8705 | 9.9322 | 0.9296 |
| | 3.0 | 507s | 1.1793 | 1.2716 | 0.1005 | 19.8262 | 9.3185 | 3.1224 |

Table 5.8 – Standard deviation of parameter estimates for the enzyme reaction network.

| $\Delta t$ $(L)$ | $\sigma^2$ | $c_1^*$ | $c_2^*$ | $c_3^*$ | $E(0)^*$ | $S(0)^*$ | $\sigma^{2,*}$ |
|---|---|---|---|---|---|---|---|
| 0.1(100) | 0.1 | 0.2212 | 0.2355 | 0.0183 | 0.0042 | 0.0050 | 0.0086 |
| | 1.0 | 0.2242 | 0.2299 | 0.0254 | 0.0296 | 0.1140 | 0.0167 |
| | 3.0 | 0.1889 | 0.3046 | 0.0255 | 0.1056 | 0.3128 | 0.0296 |
| 1.0(10) | 0.1 | 0.4105 | 0.2863 | 0.0194 | 0.0741 | 0.1226 | 0.0266 |
| | 1.0 | 0.6441 | 0.4478 | 0.0174 | 0.2736 | 0.2736 | 0.0831 |
| | 3.0 | 0.6194 | 0.7898 | 0.0207 | 0.4853 | 0.7931 | 0.1340 |

Table 5.9 – Different approximations of the standard deviations of the estimators.

| Method | $K$ | $L$ | $c_1^*$ | $c_2^*$ | $c_3^*$ | $\sigma^*$ | mRNA(0)$^*$ |
|---|---|---|---|---|---|---|---|
| Fisher inf. matrix | 10 | 10 | 0.05451 | 0.56196 | 0.93532 | 0.36434 | 0.63947 |
| 100 experiments | | | 0.03581 | 0.19870 | 0.26222 | 0.39288 | 0.49031 |
| Fisher inf. matrix | 20 | 20 | 0.03245 | 0.29949 | 0.45148 | 0.17410 | 0.59482 |
| 100 experiments | | | 0.03042 | 0.16743 | 0.28747 | 0.13451 | 0.43606 |
| Fisher inf. matrix | 50 | 50 | 0.01392 | 0.11071 | 0.15223 | 0.04403 | 0.23803 |
| 100 experiments | | | 0.01403 | 0.07852 | 0.14623 | 0.03538 | 0.18389 |
| Fisher inf. matrix | 100 | 100 | 0.00866 | 0.05483 | 0.07281 | 0.01826 | 0.20847 |
| 100 experiments | | | 0.00692 | 0.04301 | 0.06418 | 0.02176 | 0.18797 |

Table 5.10 – Chemical reactions of the multi-attractor model.

$$
\begin{aligned}
\text{PaxDna} &\xrightarrow{p} \text{PaxDna} + \text{PaxProt}\\
\text{PaxProt} &\xrightarrow{d} \emptyset\\
\text{PaxDna} + \text{DeltaProt} &\underset{u}{\overset{b}{\rightleftharpoons}} \text{PaxDnaDeltaProt}\\
\text{MafADna} &\xrightarrow{p} \text{MafADna} + \text{MafAProt}\\
\text{MafAProt} &\xrightarrow{d} \emptyset\\
\text{MafADna} + \text{PaxProt} &\underset{u}{\overset{b}{\rightleftharpoons}} \text{MafADnaPaxProt}\\
\text{MafADnaPaxProt} &\xrightarrow{p} \text{MafADnaPaxProt} + \text{MafAProt}\\
\text{MafADna} + \text{MafAProt} &\underset{u}{\overset{b}{\rightleftharpoons}} \text{MafADnaMafAProt}\\
\text{MafADnaMafAProt} &\xrightarrow{p} \text{MafADnaMafAProt} + \text{MafAProt}\\
\text{MafADna} + \text{DeltaProt} &\underset{u}{\overset{b}{\rightleftharpoons}} \text{MafADnaDeltaProt}\\
\text{DeltaDna} &\xrightarrow{p} \text{DeltaDna} + \text{DeltaProt}\\
\text{DeltaProt} &\xrightarrow{d} \emptyset\\
\text{DeltaDna} + \text{PaxProt} &\underset{u}{\overset{b}{\rightleftharpoons}} \text{DeltaDnaPaxProt}\\
\text{DeltaDnaPaxProt} &\xrightarrow{p} \text{DeltaDnaPaxProt} + \text{DeltaProt}\\
\text{DeltaDna} + \text{MafAProt} &\underset{u}{\overset{b}{\rightleftharpoons}} \text{DeltaDnaMafAProt}\\
\text{DeltaDna} + \text{DeltaProt} &\underset{u}{\overset{b}{\rightleftharpoons}} \text{DeltaDnaDeltaProt}\\
\text{DeltaDnaDeltaProt} &\xrightarrow{p} \text{DeltaDnaDeltaProt} + \text{DeltaProt}
\end{aligned}
$$

Table 5.11 – Production rate estimation in the multi-attractor model.

| protein | $K$ | estimated rate constant | standard deviation | time (hours) | observed proteins |
|---|---|---|---|---|---|
| PaxProt | 1 | 10.0 | 13.6159 | 7.45 | MafAProt, DeltaProt |
| | 5 | 0.5693 | 2.1842 | 6.34 | |
| MafAProt | 1 | 4.9998 | 4.9884 | 11.62 | PaxProt, DeltaProt |
| | 5 | 5.4853 | 2.3873 | 13.86 | |
| DeltaProt | 1 | 2.5453 | 1.8075 | 4.35 | PaxProt, MafAProt |
| | 5 | 5.3646 | 1.4682 | 12.39 | |

# 6 STAR : STochastic Analysis of biochemical Reaction networks

The popularity of software tools for stochastic modeling and simulation of biochemical reaction networks has been steadily growing over the past decade. A common interface for these tools is the Systems Biology Markup Language (SBML) ([HFS+03]) and currently over 280 software tools and packages with SBML support provide different modeling and simulation features (see [SBM16]).

Several tools were particularly designed for simulating the kinetics of biochemical reaction networks ([RDLN07], [FMJ+08]) and as a standard functionality these tools allow stochastic ([Kie02], [MOB13], [SWR+11], [CH10], [LADC09]) and deterministic simulation ([ROB05], [Mau09]). Some of the tools focus on specific analysis methods such as moment closure approximations ([Gil09], [Hes08]) or sensitivity analysis ([SRK13], [KŽS12]). The list of the tools supporting multiple analysis techniques and a comparison of their features can be found in [SBM16].

The main difference to the tool STAR is that it uses a dynamical state space representation, which is the basis for an efficient numerical approximation of stochastic models of biochemical reaction networks. This numerical solution is not based on sampling methods but on a direct solution of the corresponding master equation. The tool CERENA ([KFR+16]) also provides a direct solution of the master equation but is based on MATLAB and truncates the underlying state space using the finite state projection algorithm ([MK06]). STAR is based on the tool SHAVE ([LMW11]), which was previously developed by the same authors but not specifically designed for the solution of biochemical reaction networks Also, the tool STAR includes a higher variety of analysis methods as opposed to the


(a) (b) (c)

Figure 6.1 – Web-based graphical user interface interface of STAR (model editor (a), experiment editor (b), observing the time course (c)).

tool SHAVE which has its main focus on the transient analysis.

In Section 6.1 we provide the implementation details and describe how users can specify models of biochemical reactions in Section 6.2. Finally, in Sections 6.3 and 6.4 we describe the analysis techniques that the tool provides and how they can be used for an efficient simulation of the model.

## 6.1 Architecture

The tool STAR is available via a web-interface and as a standalone cross platform application. In order to use the web-interface a user must register. In Figure 6.1 (a)-(c) we show screenshots of the web-interface of STAR.

After logging in, the user can specify a model of a chemical reaction network using a simple language or import an SBML model. SBML import/export functionality is implemented using LibSBML [BKJH08]. Experiments can be set up using the simple scripting language Lua [IDFC11]. In the experiment script the user can choose the model that he or she wants to analyse, as well as the desired analysis techniques and output details. The output includes plots (see Figure 6.2) and data files.

The core application is implemented in C++ and, when using the standalone cross platform application, it can be run via command line. The user must provide a path to an experiment file as a command line argument. The model files must be saved locally and are set for the analysis by specifying the path to them. The simulation results are saved into the output directory and are organized by creation date and time. The

results can be conveniently viewed by opening the HTML report file in the browser. For plotting features, Gnuplot [WK+10] must be installed. For optimization and parameter estimation functionality, NLopt [Joh12] library must be installed. Also, the command line version provides SBML import/export utilities. The source code of the standalone command line version of the tool is released under the General Public License version 3 (GPLv3) and can be compiled using CMake [Mar03] under Linux and MacOS platforms. In addition, the Boost [Boo12] library is required.

## 6.2 Model specification

The tool is based on Gillespie's theory of stochastic chemical kinetics [Gil77], which considers a well-stirred mixture of $N$ molecular species $\mathbb{S} = \{S_1, \ldots, S_N\}$ interacting in a volume with fixed size and fixed temperature through $R$ chemical reactions of the form

$$\mathcal{R}_j : \; v^-_{j,1} S_1 + \ldots + v^-_{j,N} S_N \rightarrow v^+_{j,1} S_1 + \ldots + v^+_{j,N} S_N,$$

where $j \in \{1, \ldots, R\}$ and $v^-_{j,i}$, $v^+_{j,i}$ ($i \in \{1, \ldots, N\}$) are stoichiometric coefficients. The dynamics of such a network of reactions can be described by a continuous-time Markov chain $\{\mathbf{X}(t), t \geq 0\}$, where the random vector $\mathbf{X}(t) = (X_1(t), \ldots, X_N(t))$ describes the chemical populations at time $t$, i.e., $X_i(t)$ is the number of molecules of type $i \in \{1, \ldots, N\}$ at time $t$. Thus, if $\mathbf{X}(t) = \mathbf{x}$ for some $\mathbf{x} \in \mathbb{Z}^N_+$ with $\mathbf{x} + \mathbf{v}^-_j$ being non-negative, then $\mathbf{X}(t + dt) = \mathbf{x} + \mathbf{v}_j$ is the state of the system after the occurrence of the $j$-th reaction within the infinitesimal time interval $[t, t + dt)$, where $\mathbf{v}_j = \mathbf{v}^-_j + \mathbf{v}^+_j$.

For $\mathbf{x} \in \mathbb{Z}^N_+$ and $t \geq 0$, let $p(\mathbf{x}|t)$ denote the probability $Pr(\mathbf{X}(t) = \mathbf{x})$ and let $\mathbf{p}(t)$ be the row vector with entries $p(\mathbf{x}|t)$. Given some initial distribution $\mathbf{p}(0)$, the Markov chain $\mathbf{X}$ is uniquely specified and its evolution is given by the chemical master equation (CME)

$$\frac{dp(\mathbf{x}|t)}{dt} = \sum_{\substack{j=1 \\ \mathbf{x} \geq \mathbf{v}_j}}^{R} p(\mathbf{x} - \mathbf{v}_j|t)\alpha_j(\mathbf{x} - \mathbf{v}_j) - p(\mathbf{x}|t) \sum_{j=1}^{R} \alpha_j(\mathbf{x}), \tag{6.1}$$

where $\alpha_j$ is the propensity function such that $\alpha_j(\mathbf{x}) \cdot dt$ is the probability that, given $\mathbf{X}(t) = \mathbf{x}$, one instance of the $j$-th reaction occurs within $[t, t + dt)$.

If law of mass action kinetics are used, the propensity function is given as

$$\alpha_j(\mathbf{x}) = c_j \prod_{i=1}^{N} \binom{x_i}{v_{j,i}^-}, \tag{6.2}$$

where $c_j$ is the stochastic reaction rate constant and $x_i$ is the number of molecules of species $S_i$ present in state $\mathbf{x}$. Note that STAR is not limited to propensities that follow the law of mass action. In fact, the propensity function can be given as a rational function dependent on the state vector $\mathbf{x}$.

### 6.2.1 Specification language

STAR features a simple human readable language for specifying reaction networks. As an example, we consider the three-stage gene expression model [SS08, HWKT13] and specify the model using the STAR modeling language (see Listing 6.1). In lines 2-6, we define the kinetic constants. In lines 9-11, we define the variables corresponding to the chemical species of the model. Here it is important to use appropriate types such that the simulation routine can exploit constraints such as variables that can only be 0 or 1. In lines 13-21, we describe the chemical reactions and their propensities. Here, `mass_action(c)` refers to the law of mass action with a reaction rate constant $c$. Finally, in lines 24-26, we set the initial conditions, where only the initial numbers of those species must be listed that are initially present.

Also, the description of a chemical reaction can be supplied with a guard function. For example,

```
(R<100) D_on -> D_on + R  @ mass_action(k_r)
```

describes a reaction that is only possible if the number of molecules of type $R$ is less than 100.

Moreover, the transition classes can be specified in a more general way using *guarded commands* [HJW09], where each guarded command consists of a guard, an update rule and a rate function. Thus, the guarded chemical reaction above can be written as

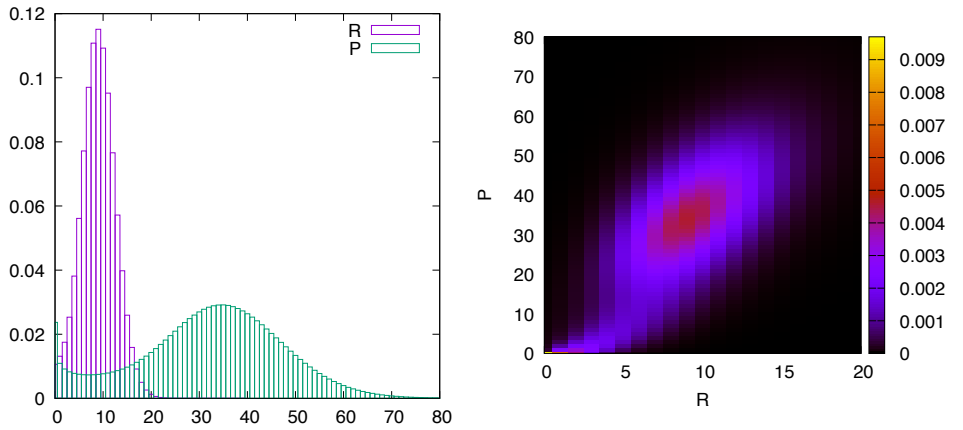
```
// kinetic constants
const
  tau_on = 1.0, tau_off = 1.0,
  k_r = 10.0, k_p = 1.0,
  gamma_r = 4.0, gamma_p = 1.0,
  tau_p_on = 0.015

// chemical species
var
  R, P : species,
  D_on, D_off : boolean

chemical_reactions
  D_off <-> D_on  @ mass_action(tau_on),
                    mass_action(tau_off)
  D_on -> D_on + R  @ mass_action(k_r)
  R -> R + P  @ mass_action(k_p)
  R -> 0  @ mass_action(gamma_r)
  P -> 0  @ mass_action(gamma_p)
  P + D_off -> P + D_on  @ mass_action(tau_p_on)
end

// initial conditions
init
  0.7: D_off = 1, R = 4, P = 10;
  0.3: D_on = 1, R = 4, P = 10;
end
```

Listing 6.1 – Three-stage gene expression model specified in STAR modeling language.




(a) (b)

Figure 6.2 – Sample output plots produced by tool STAR: distribution of mRNA and protein numbers in on-state (a), joint probability for mRNA and protein numbers in on-state (b).

```
R<100 and D_on : R′ = R + 1  @ k_r.
```

Note, that when using guarded commands, one must specify the propensity function explicitly, since the *mass_action* command can only be used in the context of chemical reactions. The formal definition of the model specification language is given in Appendix A in extended Backus–Naur form (EBNF) [Sco98].

## 6.3 Simulation methods

Exact analytical solutions of the CME are only available for very small reaction networks or special cases [JH07, Lau00] and therefore, in general, computational approaches are required. The size of the state space typically increases exponentially with the model dimensionality, that is, with the number of molecular species in the reaction network. This effect is often referred to as the curse of dimensionality or state space explosion. Furthermore, because reaction rates typically differ by several orders of magnitude, the system dynamics possess multiple time scales and the corresponding equations are stiff. Stochastic simulation of the reaction network and the numerical solution of the CME are two common complementary approaches to analyze stochastic reaction networks governed by

```
1  transient({
2    model = 'three_stage',
3    tspan = linspace(0, 10),
4    dump_moments = {
5      {vars = {'R', 'P'}, nmoments = 2},
6      {vars = {'D_on', 'D_off'}, nmoments = 3, raw = true}
7    },
8    plot = {
9      {'R', 'P'},
10     {'D_on', 'D_off'}
11   },
12   plot_distr = {
13     {'R', 'P'},
14     {vars = {'R', 'P'}, cond = 'D_on=1'}
15   },
16   plot_distr_2d = {
17     {var1 = 'R', var2 = 'P'},
18     {var1 = 'R', var2 = 'P', cond = 'D_off=1'}
19   }
20 })
```

Listing 6.2 – An experiment specification for the tool STAR.

the CME, both of which must be properly designed to cope with huge, potentially infinite multidimensional state spaces and stiffness.

The tool STAR combines several efficient methods for the approximative numerical solution of the CME. The analysis methods of the tool can be accessed by setting up an *experiment*, which can be specified using the scripting language Lua. In the experiment script the user can choose the model that he or she wants to analyse, as well as the desired analysis techniques and output details. In Listing 6.2) we show an example of an experiment for the analysis of the gene expression model specified in Section 6.2.1. In line 2 we specify the name of the model and in line 3 the time horizon of the simulation and the time points, at which we want plot the output. The following lines describe the tasks that produce plots and data files. Figure 6.2 (a) and (b) contains plots corresponding to the tasks in lines 13 and 17, respectively.

### 6.3.1 Trajectory generation

Besides the numerical simulation, STAR also provides an implementation of the stochastic simulation algorithm (SSA), which is a standard Monte-Carlo simulation approach for chemical reaction networks ([Gil76]). It

stores the frequencies of visited states for each simulation time point. This information can be further used for the plotting of probability distributions and estimation of statistical moments.

### 6.3.2 Discrete-stochastic numerical solution

STAR implements a method that approximates the solution of the CME by truncating large, possibly infinite state spaces dynamically in an iterative fashion (see Chapter 2). At a particular time instant $t$, we consider an approximation of the transient probability distribution and temporarily neglect states with a probability smaller than a positive threshold $\delta \ll 1$, that is, their probability at time $t$ is set to zero. The CME is then solved for an (adaptively chosen) time step $h$, during which the truncated state space *Sig* is adapted to the probability distribution at time $t + h$. More precisely, certain states that do not belong to the truncated part of the state space at time $t$ are added at time $t+h$, when in the meantime they receive a significant amount of probability (exceeding $\delta$). Other states whose probabilities drop below $\delta$ are temporarily neglected. The smaller the significance threshold $\delta$ is chosen the more accurate the approximation becomes. Compared to finite projection and sliding window methods ([MK06, WGMH10]), this adaptive approach is considerably more efficient since there is no a priori estimation of significant states. Instead, states that do or do not significantly contribute are detected in an on-fly-fashion.

We tested the accuracy and the performance of our software using the SBML discrete stochastic models test suite [EGW08]. We ran all tests except tests 001-19, 002-09, 002-10, 003-03, 003-04 because they contain assignment rules and/or events, which are not supported. For each test we list in Table 6.1 the maximum relative error of the mean and the standard deviation over all species and all simulation time points for which the reference solution was given. We also list the total number of states that have ever been added to *Sig* during the simulation, the average step size and the run time. For the tests we used Runge-Kutta fourth-fifth order with adaptive step size selection in combination with dynamical state space truncation. We chose $\delta = 10^{-15}$. Note that our software also contains an implementation of explicit and implicit Euler schemes with adaptive step size selection, which can be used if lower accuracy is acceptable and which perform much faster for certain models. Also, our

software includes an implementation of the fast adaptive uniformization [DHMW09, MWDH10].

### 6.3.3 Hybrid numerical solution

As an orthogonal approach to a direct solution of the CME, moment-based analysis methods have been developed [Eng06, Hes08, RMASL11, LKK09]. Instead of integrating the distribution of all states over time, the idea is to represent the distribution by its (multidimensional) statistical moments up to order $M$ and integrate the moments over time. The number of equations remains small compared to the number of states and thus the system of equations that has to be integrated is small in comparison to that of the (truncated) CME. In addition, the number of moments does not increase as the copy number of certain species increases.

Moment-based approaches, however, have a number of disadvantages. First of all, one has to reconstruct the distribution from the moments at the final time point of the integration, which is a difficult optimization problem that can only be solved for small dimensions [AMW15a, AMW15b]. Then, the moment equations often become very stiff, in particular for $M > 4$. Another problem is that the moment representation of certain systems looses information about qualitative properties such as oscillation and multistability [DMW10].

Hybrid methods have been developed to mitigate the difficulties of moment-based approaches and ensure scalability as certain chemical populations become large [Jah11, HWKT13, HMMW10, MLSH12]. The basic idea is to integrate the marginal distributions of low copy-number species according to a "small master equation" and couple this equation with the conditional moment equations for the large copy-number populations. This is a natural representation in particular for gene regulatory networks since low copy-number species represent the state of a gene (active or inactive) and thus the moment equations are integrated for all different possible gene states. The resulting system of differential equations is typically less susceptible to numerical instabilities and for increasing $M$ the total size of the system is smaller than the size of the system in a purely moment-based approach. Also, for increasing $M$ hybrid methods show greater accuracy compared to the moments obtained from a purely moment-based approach [HWKT13].

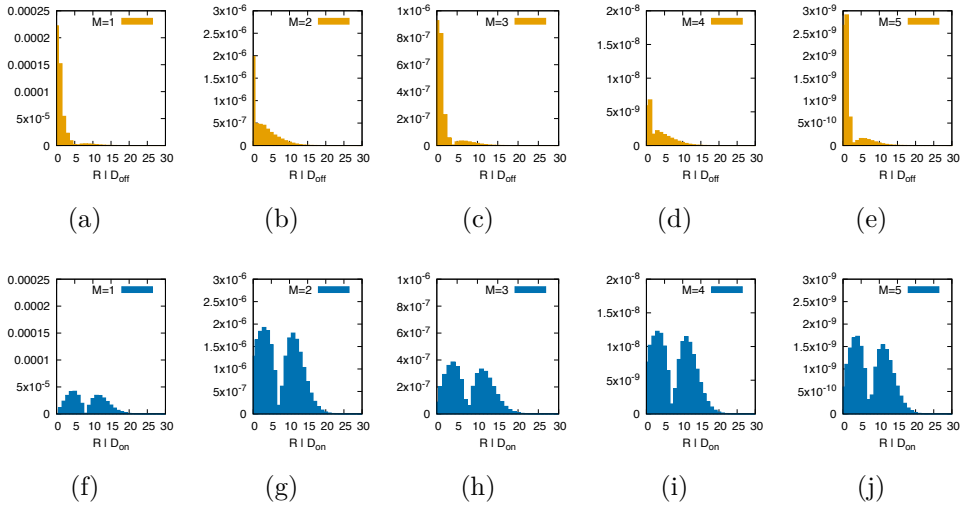



Figure 6.3 – Absolute difference between the marginal distrubution of R in on- and off-state computed using the fully stochastic numerical approach and the distribution computed using the hybrid approach $\mathrm{MCM}_{1\ldots5}(\mathrm{R})$.

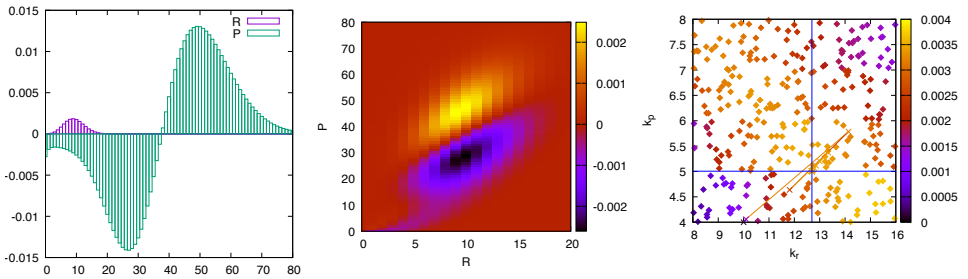
STAR implements the method of conditional moments [HWKT13, KTH14] using the dynamical state space truncation procedure, efficient numerical integration scheme, and adaptive step size selection based on local error estimates (see Chapter 3). In Figure 6.3 we plot the absolute difference between the marginal distrubution of R computed using the fully stochastic numerical approach and the distribution computed using the hybrid approach $\mathrm{MCM}_{1\ldots5}(\mathrm{R})$ in on- and off-states, where $\mathrm{MCM}_{\mathrm{M}}\left(\mathbb{S}^{(y)}\right)$ denotes the method of conditional moments for $M$ moments and for the set of discrete species $\mathbb{S}^{(y)}$.

## 6.4 Model calibration and sensitivity analysis

Based on the numerical solution of the CME, STAR provides methods for sensitivity analysis as well as parameter inference.

### 6.4.1 Sensitivity analysis

The computation of the first and second order derivatives yields sensitivities with respect to the model parameters and is carried out along with




(a) (b) (c)

Figure 6.4 – Sensitivities of the three-stage gene expression model with respect to parameter $k_p$ in one (a) and in two (b) dimensions. Parameter scanning and optimization results for the three-stage gene expression model (c).

the computation of the state probabilities using the dynamical state space representation. In Figure 6.4(a) we plot the derivatives of the marginal probability distributions over R and P molecule counts with respect to the model parameter $k_p$. In Figure 6.4(b) we plot the derivatives of the joint probability distribution over R and P molecule counts with respect to the same parameter.

### 6.4.2 Optimization

We consider an objective function of the form

$$f(T) = \sum_{\mathbf{x}} f_{\mathbf{x}} p(\mathbf{x}, T) \tag{6.3}$$

where $f_{\mathbf{x}}$ is a boolean formula that depends on the state variables and $p(\mathbf{x}, T)$ is the distribution of the model at some time point $T$. For example, let

$$f_{\mathbf{x}} = (x_{D_{on}} = 1) \vee (x_P = 50), \tag{6.4}$$

which is the indicator for being in the on-state ($x_{D_{on}} = 1$) and having 50 proteins ($x_P = 50$). Therefore, the corresponding objective function computes the probability of having 50 proteins in the on-state at time $T$.

The optimization method that is implemented in STAR allows to find the parameters of the model that optimize (e.g. maximize) the objective

function defined as (6.3). The user can choose between a number of different optimization methods available in the nonlinear optimization library NLopt. The optimization methods include derivative-based and derivative-free methods, as well as local and global optimization methods. In case of derivative-based methods, the derivatives of the objective function are efficiently computed using the sensitivities described in Section 6.4.1. In Figure 6.4(c), the zigzag vertices correspond to the intermediate points used during the optimization of (6.4) at $T = 5$ with respect to $k_r$ and $k_p$, where the starting point was chosen as $k_r = 10$ and $k_p = 4$. The intersection of blue lines correspond to the resulting point.

### 6.4.3 Parameter estimation

For given time-series experimental data, the tool estimates the parameters of the model by maximizing its likelihood given the observed data [AMSW11, AMSW12]. It is required to specify the minimum and/or maximum values of the model parameters that need to be estimated. The time-series data must be uploaded in comma separated value file format. Note that multiple observation sequences are supported. Moreover, the observables and/or the observation time points of different observation sequences do not necessarily need to match. The user can use any available optimization method mentioned in Section 6.4.2.

We assume that the structure of the underlying reaction network is known but the stochastic reaction rate constants and other parameters of the network, such as initial molecule numbers, are unknown. Then we identify the parameters of the model that maximize its likelihood given the observed data. Our algorithm for the computation of the likelihood and its derivatives w.r.t. the parameters of the model is not based on sampling but directly calculates the likelihood using a dynamic truncation of the state space. Extensive experimental results that also give insights about the identifiability of certain parameters can be found in Chapter 5.

### 6.4.4 Parameter scanning

Using the parameter scanning feature, an objective function (as defined in (6.3)) can be analyzed for different model parameters, as well as initial conditions. The parameters of the model can be iterated over predefined values and can be sampled from a number of random distributions. In

Figure 6.4(c), we plot the objective function (6.3) at $T = 5$, where the parameters $k_r$ and $k_p$ are randomly sampled 300 times from the uniform distribution on the interval $[8, 16]$ and $[4, 8]$ respectively.

Table 6.1 – Simulation results for the SBML discrete stochastic models test suite.

| test id | max. rel. err. of mean | max. rel. err. of std. dev. | total number of states | avg. step size | run time |
|---|---|---|---|---|---|
| 001-01 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-02 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-03 | 6.974e-06 | 8.733e-07 | 376 | 2.134e-03 | 5s |
| 001-04 | 6.624e-07 | 8.926e-07 | 176 | 6.203e-02 | 3s |
| 001-05 | 7.597e-08 | 2.274e-07 | 3142 | 7.685e-04 | 43s |
| 001-06 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-07 | 4.262e-07 | 3.225e-07 | 176224 | 2.769e-02 | 162s |
| 001-08 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-09 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-10 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-11 | 9.020e-08 | 2.192e-07 | 288 | 6.046e-02 | 3s |
| 001-12 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-13 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-14 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-15 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-16 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-17 | 7.588e-08 | 9.710e-07 | 339 | 2.616e-02 | 3s |
| 001-18 | 9.020e-08 | 2.192e-07 | 288 | 6.046e-02 | 3s |
| 002-01 | 2.589e-07 | 1.294e-07 | 49 | 2.415e-01 | 3s |
| 002-02 | 2.589e-07 | 1.294e-07 | 164 | 5.537e-02 | 3s |
| 002-03 | 2.589e-07 | 1.658e-06 | 119 | 9.091e-02 | 3s |
| 002-04 | 2.589e-07 | 6.091e-07 | 1450 | 7.962e-04 | 24s |
| 002-05 | 2.589e-07 | 1.294e-07 | 164 | 5.537e-02 | 3s |
| 002-06 | 3.782e-07 | 4.361e-07 | 35419 | 5.482e-02 | 15s |
| 002-07 | 2.589e-07 | 1.294e-07 | 164 | 5.537e-02 | 3s |
| 002-08 | 2.589e-07 | 1.294e-07 | 49 | 2.415e-01 | 3s |
| 003-01 | 3.762e-08 | 1.897e-07 | 51 | 2.381e-01 | 3s |
| 003-02 | 3.758e-09 | 8.998e-08 | 148 | 7.042e-02 | 3s |
| 003-05 | 3.762e-08 | 1.897e-07 | 51 | 2.381e-01 | 3s |
| 003-06 | 3.762e-08 | 1.897e-07 | 51 | 2.381e-01 | 3s |
| 003-07 | 3.762e-08 | 1.897e-07 | 51 | 2.381e-01 | 3s |
| 004-01 | 3.819e-08 | 1.016e-07 | 151 | 6.203e-02 | 3s |
| 004-02 | 2.015e-08 | 4.265e-08 | 226 | 2.329e-02 | 3s |
| 004-03 | 1.445e-07 | 1.142e-07 | 1215 | 5.509e-04 | 30s |

# 7 Conclusions

In Chapter 2, we have shown that our numerical integration approach with adaptive step size selection based on local error estimates performed in combination with dynamical state space truncation provides a versatile means of approximating the solution of the chemical master equation for complex stochastic reaction networks efficiently and accurately. The state space explosion problem is circumvented by considering in each time step only such states of the overall state space of the reaction network that have at that time step a significant (sufficiently large according to a flexibly adjustable bound) probability, that is, in the course of the integration scheme we keep the number of differential equations to be integrated per step manageable. By a framework that includes explicit as well as implicit integration schemes we offer the flexibility to choose an appropriate integration scheme that is well-suited with regard to the specific dynamics of a given reaction network. In order to provide meaningful, detailed comparisons of different methods with different parameter choices and to study their impact on accuracy, run times and numbers of significant states to be processed we have considered the explicit Euler method, an explicit Runge-Kutta as an extension of explicit Euler, and the implicit (backward) Euler method, all equipped with a well-suited adaptive step size selection strategy and performed on the dynamically truncated state space. The results show that the proposed approximate numerical integration of the chemical master equation indeed yields satisfactorily accurate results in reasonable time. Future research will be concerned with further advanced integration schemes, to equip them similarly with adaptive step size selection strategies and to study the accuracy and the run times. Of course, also in-depth theoretical investigations of the performance, efficiency and accuracy of the approximate numerical integration approach are highly

desirable.

In Chapter 3, we combined the dynamical state space truncation with the method of conditional moments, which allowed us to convert the system of differential algebraic equations into a system of ordinary differential equations. We presented the implementation details of our method and demonstrated its accuracy using two biochemical reaction networks, one of which can only be numerically analyzed using the presented method presently.

In Chapter 4, we presented an accurate and computationally efficient numerical method for approximating rare event probabilities in stochastic models of biochemically reacting systems and Markovian queueing networks with huge or infinite state space. Rather than estimating such probabilities via stochastic simulation, we numerically integrate the chemical master equation. Our methods combines a dynamical state space truncation procedure with the change of measure idea of importance sampling. This combination yields a guided state space exploration where the change of measure is applied in order to guide the analysis algorithm to the relevant parts of the state space and to avoid truncation of important states. Our method has the general advantages of numerical methods over stochastic simulation that it does not require the generation of Markov chain trajectories and has only a numerical error but no statistical error. Moreover, our experimental results show that it is not very sensitive to the specific parameter biasing, that is, our method performs well for a quite broad range of the biasing factors. This is a significant advantage over weighted stochastic simulation algorithms, which are known to require very specific biasing parameters in order to estimate rare event probabilities efficiently and with high statistical accuracy. Obtaining such biasing parameters for weighted stochastic simulation algorithms by hand is intricate and despite recent advances in automated parameter selection via the cross-entropy method still often the determination of suitable biasing parameters takes a substantial amount of computational time.

In Chapter 5, we proposed an efficient numerical method to approximate maximum likelihood estimators for a given set of observations. We consider the case where the observations are subject to measurement errors and where only the molecule numbers of some of the chemical species are observed at certain points in time. In our experiments we show that if the observations provide sufficient information then parameters can

be accurately identified. If only little information is available then the approximations of the standard deviations of the estimators indicate whether more observations are necessary to accurately calibrate certain parameters.

In Chapter 6, we presented a software tool, which provides a variety of efficient numerical methods for the analysis of stochastic models of chemical reaction networks. Models can be imported from SBML or can be specified using a simple human readable language. The tool is available for use via a user-friendly web-interface and as a standalone cross platform application. Its striking feature is the powerful numerical engine that integrates the underlying master equation or the corresponding hybrid moment equations of the stochastic model. In this way, all probabilities of interest can be directly computed including likelihoods for measured data. Hence, parameter inference tasks and sensitivity analyses are very efficient and give accurate results.

# A Model specification language

⟨*model-description*⟩ ::= {⟨*type-definitions*⟩} {⟨*constant-definitions*⟩} {⟨*variable-definitions*⟩} {⟨*function-definitions*⟩} {⟨*transition-classes*⟩} [⟨*init-section*⟩]

⟨*transition-classes*⟩ ::= {⟨*chemical-reactions*⟩} {⟨*guarded-commands*⟩}

⟨*chemical-reactions*⟩ ::= '`chemical_reactions`' {⟨*chemical-reaction*⟩} '`end`'

⟨*chemical-reaction*⟩ ::= [ '`[`' ⟨*action-identifier*⟩ '`]`' ] [ '`(`' ⟨*guard*⟩ '`)`' ] ⟨*reactants*⟩ ⟨*arrow*⟩ ⟨*products*⟩ '`@`' ⟨*rate*⟩ [ '`;`' ]

⟨*arrow*⟩ ::= '`->`' | '`<->`'

⟨*reactants*⟩ ::= '`0`' | ⟨*reactant*⟩ { '`+`' ⟨*reactant*⟩ }

⟨*reactant*⟩ ::= [ ⟨*integer-literal*⟩ ] ⟨*variable-identifier*⟩

⟨*products*⟩ ::= ⟨*reactants*⟩

⟨*guarded-commands*⟩ ::= '`guarded_commands`' {⟨*guarded-command*⟩} '`end`'

⟨*guarded-command*⟩ ::= [ '`[`' ⟨*action-identifier*⟩ '`]`' ] ⟨*guard*⟩ '`:`' ⟨*updates*⟩ '`@`' ⟨*rate*⟩ [ '`;`' ]

⟨*updates*⟩ ::= '`true`' | ⟨*update*⟩ { '`and`' ⟨*update*⟩ }

⟨*update*⟩ ::= ⟨*variable-identifier*⟩'`'`' '`=`' ⟨*expression*⟩

⟨*type-definitions*⟩ ::= ‘`type`’ ⟨*type-definition*⟩ { ‘`,`’
⟨*type-definition*⟩ } [ ‘`;`’ ]

⟨*type-definition*⟩ ::= ⟨*type-identifier*⟩ ‘`=`’ ⟨*ordinal-type*⟩

⟨*constant-definitions*⟩ ::= ‘`const`’ ⟨*constant-definition*⟩ { ‘`,`’
⟨*constant-definition*⟩ } [ ‘`;`’ ]

⟨*constant-definition*⟩ ::= ⟨*constant-identifier*⟩ [ ‘`:`’ ⟨*type*⟩ ] ‘`=`’
⟨*constant-expression*⟩

⟨*variable-definitions*⟩ ::= ‘`var`’ ⟨*variable-definition*⟩ { ‘`,`’
⟨*variable-definition*⟩ } [ ‘`;`’ ]

⟨*variable-definition*⟩ ::= ⟨*variable-identifier*⟩ { ‘`,`’ ⟨*variable-identifier*⟩ }
[ ‘`:`’ ⟨*ordinal-type*⟩ ]

⟨*function-definitions*⟩ ::= ‘`function`’ ⟨*function-definition*⟩ { ‘`,`’
⟨*function-definition*⟩ } [ ‘`;`’ ]

⟨*function-definition*⟩ ::= ⟨*function-identifier*⟩ [ ‘`(`’ ⟨*function-arguments*⟩
‘`)`’ ] [ ‘`:`’ ⟨*type*⟩ ] ‘`=`’ ⟨*function-return-expression*⟩

⟨*function-arguments*⟩ ::= ⟨*function-argument*⟩ { ‘`,`’
⟨*function-argument*⟩
} [ ‘`;`’ ]

⟨*function-argument*⟩ ::= ⟨*function-argument-identifier*⟩ { ‘`,`’
⟨*function-argument-identifier*⟩ } [ ‘`:`’ ⟨*type*⟩ ]

⟨*init-section*⟩ ::= ‘`init`’ ( ⟨*initial-state*⟩ | ⟨*initial-distribution*⟩ ) ‘`end`’

⟨*initial-state*⟩ ::= ⟨*variable-initialization*⟩ { ‘`,`’
⟨*variable-initialization*⟩ } [ ‘`;`’ ]

⟨*initial-distribution*⟩ ::= { ⟨*probability*⟩ ‘`:`’ ⟨*initial-state*⟩ }

⟨*probability*⟩ ::= ‘`0`’ | ‘`1`’ | ⟨*floating-literal*⟩

⟨*variable-initialization*⟩ ::= ⟨*variable-identifier*⟩ ‘`=`’ ⟨*constant-expression*⟩

⟨*function-return-expression*⟩ ::= ⟨*expression*⟩

⟨*guard*⟩ ::= ⟨*expression*⟩

⟨*rate*⟩ ::= ⟨*expression*⟩

⟨*expression*⟩ ::= ⟨*conditional-expression*⟩

⟨*constant-expression*⟩ ::= ⟨*conditional-expression*⟩

⟨*conditional-expression*⟩ ::= ⟨*or-expression*⟩
| ⟨*or-expression*⟩ '`?`' ⟨*expression*⟩ '`:`' ⟨*expression*⟩

⟨*or-expression*⟩ ::= ⟨*and-expression*⟩ | ⟨*or-expression*⟩ '`or`'
⟨*and-expression*⟩

⟨*and-expression*⟩ ::= ⟨*equality-expression*⟩ | ⟨*and-expression*⟩ '`and`'
⟨*or-expression*⟩

⟨*equality-expression*⟩ ::= ⟨*relational-expression*⟩
| ⟨*equality-expression*⟩ '`=`' ⟨*equality-expression*⟩
| ⟨*equality-expression*⟩ '`<>`' ⟨*equality-expression*⟩

⟨*relational-expression*⟩ ::= ⟨*additive-expression*⟩
| ⟨*relational-expression*⟩ '`<`' ⟨*relational-expression*⟩
| ⟨*relational-expression*⟩ '`<=`' ⟨*relational-expression*⟩
| ⟨*relational-expression*⟩ '`>=`' ⟨*relational-expression*⟩
| ⟨*relational-expression*⟩ '`>`' ⟨*relational-expression*⟩

⟨*additive-expression*⟩ ::= ⟨*multiplicative-expression*⟩
| ⟨*additive-expression*⟩ '`+`' ⟨*multiplicative-expression*⟩
| ⟨*additive-expression*⟩ '`-`' ⟨*multiplicative-expression*⟩

⟨*multiplicative-expression*⟩ ::= ⟨*unary-expression*⟩
| ⟨*multiplicative-expression*⟩ '`*`' ⟨*unary-expression*⟩
| ⟨*multiplicative-expression*⟩ '`/`' ⟨*unary-expression*⟩

⟨*unary-expression*⟩ ::= [⟨*unary-operator*⟩] ⟨*primary-expression*⟩

⟨*unary-operator*⟩ ::= '`+`' | '`-`' | '`not`'

⟨*primary-expression*⟩ ::= ⟨*literal*⟩ | ⟨*identifier*⟩ | '`(`' ⟨*expression*⟩ '`)`'

⟨*type*⟩ ::= ⟨*simple-type*⟩ | ⟨*type-identifier*⟩

⟨*simple-type*⟩ ::= ⟨*ordinal-type*⟩ | '`real`'

⟨*ordinal-type*⟩ ::= '`boolean`' | '`integer`' | '`species`' | ⟨*subrange-type*⟩

⟨*subrange-type*⟩ ::= ⟨*min-value*⟩ '`..`' ⟨*max-value*⟩

⟨*min-value*⟩ ::= ⟨*constant-expression*⟩

⟨*max-value*⟩ ::= ⟨*constant-expression*⟩

⟨*type-identifier*⟩ ::= ⟨*identifier*⟩

⟨*constant-identifier*⟩ ::= ⟨*identifier*⟩

⟨*variable-identifier*⟩ ::= ⟨*identifier*⟩

⟨*function-identifier*⟩ ::= ⟨*identifier*⟩

⟨*function-argument-identifier*⟩ ::= ⟨*identifier*⟩

⟨*action-identifier*⟩ ::= ⟨*identifier*⟩

⟨*identifier*⟩ ::= ⟨*alpha-char*⟩ { '`_`' | ⟨*alpha-char*⟩ | ⟨*digit*⟩ }

⟨*literal*⟩ ::= ⟨*boolean-literal*⟩ | ⟨*integer-literal*⟩ | ⟨*floating-literal*⟩

⟨*boolean-literal*⟩ ::= '`true`' | '`false`'

⟨*integer-literal*⟩ ::= ⟨*digit-seq*⟩

⟨*floating-literal*⟩ ::= ⟨*digit-seq*⟩ '`.`' [⟨*digit-seq*⟩] [⟨*exp-part*⟩]
| ⟨*digit-seq*⟩ ⟨*exp-part*⟩

⟨*exp-part*⟩ ::= ( '`e`' | '`E`' ) [⟨*sign*⟩] ⟨*digit-seq*⟩

⟨*sign*⟩ ::= '`+`' | '`-`'

⟨*alpha-char*⟩ ::= '`A`' | '`B`' | '`C`' | '`D`' | '`E`' | '`F`' | '`G`' | '`H`' | '`I`' | '`J`' | '`K`' | '`L`' |
'`M`' | '`N`' | '`O`' | '`P`' | '`Q`' | '`R`' | '`S`' | '`T`' | '`U`' | '`V`' | '`W`' | '`X`' | '`Y`' | '`Z`' | '`a`' |
'`b`' | '`c`' | '`d`' | '`e`' | '`f`' | '`g`' | '`h`' | '`i`' | '`j`' | '`k`' | '`l`' | '`m`' | '`n`' | '`o`' | '`p`' |
'`q`' | '`r`' | '`s`' | '`t`' | '`u`' | '`v`' | '`w`' | '`x`' | '`y`' | '`z`'

⟨*digit-seq*⟩ ::= ⟨*digit*⟩ {⟨*digit*⟩}

⟨*digit*⟩ ::= '`0`' | '`1`' | '`2`' | '`3`' | '`4`' | '`5`' | '`6`' | '`7`' | '`8`' | '`9`'

# Bibliography


[AG07] Søren Asmussen and Peter W Glynn. *Stochastic simulation: algorithms and analysis*, volume 57. Springer Science & Business Media, 2007.

[AKS13] Angelique Ale, Paul Kirk, and Michael PH Stumpf. A general moment expansion method for stochastic kinetic models. *The Journal of chemical physics*, 138(17):174101, 2013.

[AMSW11] Aleksandr Andreychenko, Linar Mikeev, David Spieler, and Verena Wolf. Parameter identification for markov models of biochemical reactions. In *Computer Aided Verification*, pages 83–98. Springer, 2011.

[AMSW12] Aleksandr Andreychenko, Linar Mikeev, David Spieler, and Verena Wolf. Approximate maximum likelihood estimation for stochastic chemical kinetics. *EURASIP Journal on Bioinformatics and Systems Biology*, 2012(1):1–14, 2012.

[AMW15a] Alexander Andreychenko, Linar Mikeev, and Verena Wolf. Model reconstruction for moment-based stochastic chemical kinetics. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 25(2):12, 2015.

[AMW15b] Alexander Andreychenko, Linar Mikeev, and Verena Wolf. Reconstruction of multimodal distributions for hybrid moment-based chemical kinetics. *Journal of Coupled Systems and Multiscale Dynamics*, 3(2):156–163, 2015.

[And08] David F Anderson. Incorporating postleap checks in tau-leaping. *The Journal of chemical physics*, 128(5):054103, 2008.

[BHM+06] Kevin Burrage, MARKUS Hegland, Shev Macnamara, Roger Sidje, et al. A krylov-based finite state projection algorithm for solving the chemical master equation arising in the discrete modelling of biological systems. In *Proc. of The AA Markov 150th Anniversary Meeting*, pages 21–37, 2006.

[BKJH08] Benjamin J Bornstein, Sarah M Keating, Akiya Jouraku, and Michael Hucka. Libsbml: an api library for sbml. *Bioinformatics*, 24(6):880–881, 2008.

[BKL75] Alfred B Bortz, Malvin H Kalos, and Joel L Lebowitz. A new algorithm for monte carlo simulation of ising spin systems. *Journal of Computational Physics*, 17(1):10–18, 1975.

[Boo12] C++ Boost. Libraries, 2012.

[BSW06] Hauke Busch, Werner Sandmann, and Verena Wolf. A numerical aggregation algorithm for the enzyme-catalyzed substrate conversion. In *International Conference on Computational Methods in Systems Biology*, pages 298–311. Springer, 2006.

[BTB04] Kevin Burrage, Tianhai Tian, and Pamela Burrage. A multi-scaled approach for simulating chemical reaction systems. *Progress in biophysics and molecular biology*, 85(2):217–234, 2004.

[Buc13] James Bucklew. *Introduction to rare event simulation*. Springer Science & Business Media, 2013.

[But08] John C Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2008.

[BWK08] Richard J Boys, Darren J Wilkinson, and Thomas BL Kirkwood. Bayesian inference for a discretely observed stochastic kinetic model. *Statistics and Computing*, 18(2):125–135, 2008.

[CGP06] Yang Cao, Daniel T Gillespie, and Linda R Petzold. Efficient step size selection for the tau-leaping simulation method. *The Journal of chemical physics*, 124(4):044109, 2006.

[CGP07] Yang Cao, Daniel T Gillespie, and Linda R Petzold. Adaptive explicit-implicit tau-leaping method with automatic tau selection. *The Journal of chemical physics*, 126(22):224101, 2007.

[CH10] Emmet Caulfield and Andreas Hellander. Cellmc—a multiplatform model compiler for the cell broadband engine and× 86. *Bioinformatics*, 26(3):426–428, 2010.

[CLP04] Yang Cao, Hong Li, and Linda Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *The journal of chemical physics*, 121(9):4059–4067, 2004.

[CNY08] Ching-Shan Chou, Qing Nie, and Tau-Mu Yi. Modeling robustness tradeoffs in yeast cell polarization induced by spatial gradients. *PloS one*, 3(9):e3103, 2008.

[DB06] Pieter-Tjerk De Boer. Analysis of state-independent importance-sampling measures for the two-node tandem queue. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 16(3):225–250, 2006.

[DHMW09] Frederic Didier, Thomas A Henzinger, Maria Mateescu, and Verena Wolf. Fast adaptive uniformization of the chemical master equation. In *High Performance Computational Systems Biology, 2009. HIBI'09. International Workshop on*, pages 118–127. IEEE, 2009.

[DJRGP11] Bernie J Daigle Jr, Min K Roh, Dan T Gillespie, and Linda R Petzold. Automated estimation of rare event probabilities in biochemical systems. *The Journal of chemical physics*, 134(4):044110, 2011.

[DLW07] Paul Dupuis, Kevin Leder, and Hui Wang. Large deviations and importance sampling for a tandem network with slowdown. *Queueing Systems*, 57(2-3):71–83, 2007.

[DMW10] Tugrul Dayar, Linar Mikeev, and Verena Wolf. On the numerical analysis of stochastic lotka-volterra models. In *Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on*, pages 289–296. IEEE, 2010.

[DP80] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

[dSeSO92] Edmundo de Souza e Silva and Pedro Mejiá Ochoa. State space exploration in markov models. In *ACM SIGMETRICS Performance Evaluation Review*, volume 20, pages 152–166. ACM, 1992.

[EGW08] Thomas W Evans, Colin S Gillespie, and Darren J Wilkinson. The sbml discrete stochastic models test suite. *Bioinformatics*, 24(2):285–286, 2008.

[Eng06] Stefan Engblom. Computing the moments of high dimensional solutions of the master equation. *Applied Mathematics and Computation*, 180(2):498–515, 2006.

[FMJ+08] Akira Funahashi, Yukiko Matsuoka, Akiya Jouraku, Mineo Morohashi, Norihiro Kikuchi, and Hiroaki Kitano. Celldesigner 3.5: a versatile modeling tool for biochemical networks. *Proceedings of the IEEE*, 96(8):1254–1265, 2008.

[GB00] Michael A Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The journal of physical chemistry A*, 104(9):1876–1889, 2000.

[GG10] Colin S Gillespie and Andrew Golightly. Bayesian inference for generalized stochastic population growth models with application to aphids. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 59(2):341–357, 2010.

[Gil76] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics*, 22(4):403–434, 1976.

[Gil77] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

[Gil92] Daniel T Gillespie. A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188(1):404–425, 1992.

[Gil01] Daniel T Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[Gil09] Colin S Gillespie. Moment-closure approximations for mass-action models. *IET systems biology*, 3(1):52–58, 2009.

[GK95] Paul Glasserman and Shing-Gang Kou. Analysis of an importance sampling estimator for tandem queues. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 5(1):22–42, 1995.

[GPZC05] Ido Golding, Johan Paulsson, Scott M Zawilski, and Edward C Cox. Real-time kinetics of gene activity in individual bacteria. *Cell*, 123(6):1025–1036, 2005.

[GRP09] Dan T Gillespie, Min Roh, and Linda R Petzold. Refining the weighted stochastic simulation algorithm. *The Journal of chemical physics*, 130(17):174103, 2009.

[HC06] Leonard A Harris and Paulette Clancy. A "partitioned leaping" approach for multiscale modeling of chemical reaction dynamics. *The Journal of chemical physics*, 125(14):144107, 2006.

[Hes08] Joao Hespanha. Moment closure for biochemical networks. In *Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on*, pages 142–147. IEEE, 2008.

[HFS+03] Michael Hucka, Andrew Finney, Herbert M Sauro, Hamid Bolouri, John C Doyle, Hiroaki Kitano, Adam P Arkin, Benjamin J Bornstein, Dennis Bray, Athel Cornish-Bowden, et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[Hig77] JJ Higgins. Bayesian inference and the optimality of maximum likelihood estimation. *International Statistical Review/Revue Internationale de Statistique*, 45(1):9–11, 1977.

[HJW09] Thomas A Henzinger, Barbara Jobstmann, and Verena Wolf. Formalisms for specifying markovian population models. In

*International Workshop on Reachability Problems*, pages 3–23. Springer, 2009.

[HL07] Andreas Hellander and Per Lötstedt. Hybrid method for the chemical master equation. *Journal of Computational Physics*, 227(1):100–122, 2007.

[HMMW10] Thomas A Henzinger, Linar Mikeev, Maria Mateescu, and Verena Wolf. Hybrid numerical solution of the chemical master equation. In *Proceedings of the 8th International Conference on Computational Methods in Systems Biology*, pages 55–65. ACM, 2010.

[HNW93] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag, Berlin Heidelberg, 2nd edition, 1993.

[HR02] Eric L Haseltine and James B Rawlings. Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics. *The Journal of chemical physics*, 117(15):6959–6969, 2002.

[HW96] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer-Verlag, Berlin Heidelberg, 2nd edition, 1996.

[HWKT13] J Hasenauer, V Wolf, A Kazeroonian, and FJ Theis. Method of conditional moments (mcm) for the chemical master equation. *Journal of mathematical biology*, pages 1–49, 2013.

[IDFC11] Roberto Ierusalimschy, Luiz Henrique De Figueiredo, and Waldemar Celes. Lua 5.2 reference manual, 2011.

[Jah10] Tobias Jahnke. An adaptive wavelet method for the chemical master equation. *SIAM Journal on Scientific Computing*, 31(6):4373–4394, 2010.

[Jah11] Tobias Jahnke. On reduced models for the chemical master equation. *Multiscale Modeling & Simulation*, 9(4):1646–1676, 2011.

[JH07] Tobias Jahnke and Wilhelm Huisinga. Solving the chemical master equation for monomolecular reaction systems

analytically. *Journal of mathematical biology*, 54(1):1–26, 2007.

[Joh12] Steven G Johnson. The nlopt nonlinear-optimization package (version 2.4.2). *URL http://ab-initio. mit. edu/nlopt*, 2012.

[JU10] Tobias Jahnke and Tudor Udrescu. Solving chemical master equations by adaptive wavelet compression. *Journal of Computational Physics*, 229(16):5724–5741, 2010.

[KFHR09] Michał Komorowski, Bärbel Finkenstädt, Claire V Harper, and David A Rand. Bayesian inference of biochemical kinetic parameters using the linear noise approximation. *BMC bioinformatics*, 10(1):1, 2009.

[KFR+16] A Kazeroonian, F Fröhlich, A Raue, FJ Theis, and J Hasenauer. Cerena: Chemical reaction network analyzer-a toolbox for the simulation and analysis of stochastic chemical kinetics. *PloS one*, 11(1):e0146732, 2016.

[Kie02] Andrzej M Kierzek. Stocks: Stochastic kinetic simulations of biochemical systems with gillespie algorithm. *Bioinformatics*, 18(3):470–481, 2002.

[KM08] Hiroyuki Kuwahara and Ivan Mura. An efficient and exact stochastic simulation method to analyze rare events in biochemical systems. *The Journal of chemical physics*, 129(16):165101, 2008.

[KTB13] Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of monte carlo methods*, volume 706. John Wiley & Sons, 2013.

[KTH14] Atefeh Kazeroonian, Fabian J Theis, and Jan Hasenauer. Modeling of stochastic biological processes with non-polynomial propensities using non-central conditional moment equation. 2014.

[Kur72] Thomas G Kurtz. The relationship between stochastic and deterministic models for chemical reactions. *The Journal of Chemical Physics*, 57(7):2976–2978, 1972.

[KŽS12] Michał Komorowski, Justina Žurauskienė, and Michael PH Stumpf. Stochsens—matlab package for sensitivity analysis of stochastic chemical systems. *Bioinformatics*, 28(5):731–733, 2012.

[LADC09] Mieszko Lis, Maxim N Artyomov, Srinivas Devadas, and Arup K Chakraborty. Efficient stochastic simulation of reaction–diffusion processes via direct compilation. *Bioinformatics*, 25(17):2289–2291, 2009.

[Lau00] Ian J Laurenzi. An analytical solution of the stochastic master equation for reversible bimolecular reaction kinetics. *The Journal of Chemical Physics*, 113(8):3315–3322, 2000.

[LBTG10] Pierre L'ecuyer, Jose H Blanchet, Bruno Tuffin, and Peter W Glynn. Asymptotic robustness of estimators in rare-event simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 20(1):6, 2010.

[Lju99] Lennart Ljung. System identification: Theory for the user, ptr prentice hall information and system sciences series, 1999.

[LKK09] Chang Hyeong Lee, Kyeong-Hun Kim, and Pilwon Kim. A moment closure method for stochastic reaction networks. *The Journal of chemical physics*, 130(13):134107, 2009.

[LLBB07] Adiel Loinger, Azi Lipshtat, Nathalie Q Balaban, and Ofer Biham. Stochastic simulations of genetic switch systems. *Physical Review E*, 75(2):021904, 2007.

[LMW11] Maksim Lapin, Linar Mikeev, and Verena Wolf. Shave: stochastic hybrid analysis of markov population models. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pages 311–312. ACM, 2011.

[LP06] Hong Li and Linda Petzold. Logarithmic direct method for discrete stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 2006.

[Mar03] Ken Martin. The cmake build manager. *Dr. Dobb's Journal of Software Tools*, 28(1):40, 2003.

[MAT15] Global Optimization Toolbox MATLAB. User's guide (r2015b), 2015.

[Mau09] S Mauch. Cain: Stochastic simulations for chemical kinetics. *URL: http://cain. sourceforge. net*, 2009.

[MBBS08] Shev MacNamara, Alberto M Bersani, Kevin Burrage, and Roger B Sidje. Stochastic chemical kinetics and the total quasi-steady-state assumption: application to the stochastic simulation algorithm and chemical master equation. *The Journal of chemical physics*, 129(9):095105, 2008.

[MBS08] Shev MacNamara, Kevin Burrage, and Roger B Sidje. Multiscale modeling of chemical kinetics via the master equation. *Multiscale Modeling & Simulation*, 6(4):1146–1168, 2008.

[Mir09] Denis Miretskiy. *Queueing networks: rare events and fast simulations*. University of Twente, 2009.

[MK06] Brian Munsky and Mustafa Khammash. The finite state projection algorithm for the solution of the chemical master equation. *The Journal of chemical physics*, 124(4):044104, 2006.

[MK07] Brian Munsky and Mustafa Khammash. A multiple time interval finite state projection algorithm for the solution to the chemical master equation. *Journal of Computational Physics*, 226(1):818–835, 2007.

[MLSH12] Stephan Menz, Juan C Latorre, Christof Schütte, and Wilhelm Huisinga. Hybrid stochastic–deterministic solution of the chemical master equation. *Multiscale Modeling & Simulation*, 10(4):1232–1262, 2012.

[MOB13] Timo R Maarleveld, Brett G Olivier, and Frank J Bruggeman. Stochpy: A comprehensive, user-friendly tool for simulating stochastic biological processes. *PloS one*, 8(11):e79345, 2013.

[MPC+06] James M McCollum, Gregory D Peterson, Chris D Cox, Michael L Simpson, and Nagiza F Samatova. The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. *Computational biology and chemistry*, 30(1):39–49, 2006.

[MSM07] Denis I Miretskiy, Werner RW Scheinhardt, and MRH Mandjes. Efficient simulation of a tandem queue with server slow-down. *Simulation*, 83(11):751–767, 2007.

[MVL78] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM review*, 20(4):801–836, 1978.

[MVL03] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review*, 45(1):3–49, 2003.

[MW12] Linar Mikeev and Verena Wolf. Parameter estimation for stochastic hybrid models of biochemical reaction networks. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pages 155–166. ACM, 2012.

[MWDH10] M Mateescu, V Wolf, F Didier, and TA Henzinger. Fast adaptive uniformisation of the chemical master equation. *Systems Biology, IET*, 4(6):441–452, 2010.

[PB00] David Pruyne and Anthony Bretscher. Polarization of cell growth in yeast. *Journal of Cell Science*, 113(4):571–585, 2000.

[PW89] Shyam Parekh and Jean Walrand. A quick simulation method for excessive backlogs in networks of queues. *Automatic Control, IEEE Transactions on*, 34(1):54–66, 1989.

[RA03] Christopher V Rao and Adam P Arkin. Stochastic chemical kinetics and the quasi-steady-state assumption: application to the gillespie algorithm. *The Journal of chemical physics*, 118(11):4999–5010, 2003.

[RAT06] S Reinker, RM Altman, and J Timmer. Parameter estimation in stochastic biochemical reactions. *IEE Proceedings-Systems Biology*, 153(4):168–178, 2006.

[RDJGP11] Min K Roh, Bernie J Daigle Jr, Dan T Gillespie, and Linda R Petzold. State-dependent doubly weighted stochastic simulation algorithm for automatic characterization of stochastic biochemical rare events. *The Journal of chemical physics*, 135(23):234108, 2011.

[RDLN07] Nicolas Rodriguez, Marco Donizelli, and Nicolas Le Novère. Sbmleditor: effective creation of models in the systems biology markup language (sbml). *BMC bioinformatics*, 8(1):79, 2007.

[RES07] Muruhan Rathinam and Hana El Samad. Reversible-equivalent-monomolecular tau: A leaping method for "small number and stiff" stochastic chemical systems. *Journal of Computational Physics*, 224(2):897–923, 2007.

[RK13] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[RMASL11] J Ruess, A Milias-Argeitis, S Summers, and J Lygeros. Moment estimation for chemically reacting systems by extended kalman filtering. *The Journal of chemical physics*, 135(16):165102, 2011.

[ROB05] Stephen Ramsey, David Orrell, and Hamid Bolouri. Dizzy: stochastic simulation of large-scale genetic regulatory networks. *Journal of bioinformatics and computational biology*, 3(02):415–436, 2005.

[RPCG03] Muruhan Rathinam, Linda R Petzold, Yang Cao, and Daniel T Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *The Journal of Chemical Physics*, 119(24):12784–12794, 2003.

[RT+09] Gerardo Rubino, Bruno Tuffin, et al. *Rare event simulation using Monte Carlo methods*, volume 73. Wiley Online Library, 2009.

[San04a] Werner Sandmann. Fast simulation of excessive population size in tandem jackson networks. In *Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004.(MASCOTS 2004). Proceedings. The IEEE Computer Society's 12th Annual International Symposium on*, pages 347–354. IEEE, 2004.

[San04b] Werner Sandmann. Structured description of markovian network models and its potentials for efficient rare event simulation. In *Proceedings of the 2nd International Conference on Performance Modelling and Evaluation of Heterogeneous Networks, HetNets*, volume 4, page P39. Citeseer, 2004.

[San07] Werner Sandmann. Efficiency of importance sampling estimators&star. *Journal of Simulation*, 1(2):137–145, 2007.

[San08] Werner Sandmann. Discrete-time stochastic modeling and simulation of biochemical networks. *Computational biology and chemistry*, 32(4):292–297, 2008.

[San09a] Werner Sandmann. Rare event simulation methodologies in systems biology. *Rare Event Simulation Using Monte Carlo Methods*, pages 243–266, 2009.

[San09b] Werner Sandmann. Sequential estimation for prescribed statistical accuracy in stochastic simulation of biological systems. *Mathematical biosciences*, 221(1):43–53, 2009.

[San09c] Werner Sandmann. Streamlined formulation of adaptive explicit-implicit tau-leaping with automatic tau selection. In *Winter Simulation Conference*, pages 1104–1112. Winter Simulation Conference, 2009.

[SBM16] SBML.org. Sbml software guide, 2016. http://sbml.org/SBML_Software_Guide, Accessed on 31 March 2016.

[Sco98] Roger S Scowen. Extended bnf-a generic base standard. Technical report, Technical report, ISO/IEC 14977. http://www.cl. cam. ac. uk/mgk25/iso-14977. pdf, 1998.

[SGT03] Lawrence F Shampine, Ian Gladwell, and Skip Thompson. *Solving ODEs with matlab*. Cambridge University Press, 2003.

[SK05] Howard Salis and Yiannis Kaznessis. Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *The Journal of chemical physics*, 122(5):054103, 2005.

[SPA05] Michael Samoilov, Sergey Plyasunov, and Adam P Arkin. Stochastic amplification and signaling in enzymatic futile cycles through noise-induced bistability with oscillations. *Proceedings of the National Academy of Sciences of the United States of America*, 102(7):2310–2315, 2005.

[SRK13] Patrick W Sheppard, Muruhan Rathinam, and Mustafa Khammash. Spsens: A software package for stochastic parameter sensitivity analysis of biochemical reaction networks. *Bioinformatics*, 29(1):140–142, 2013.

[SS08] Vahid Shahrezaei and Peter S Swain. Analytical distributions for stochastic gene expression. *Proceedings of the National Academy of Sciences*, 105(45):17256–17261, 2008.

[SSDN15] Cosmin Safta, Khachik Sargsyan, Bert Debusschere, and Habib N Najm. Hybrid discrete/continuum algorithms for stochastic reaction networks. *Journal of Computational Physics*, 281:177–198, 2015.

[SW08] Werner Sandmann and Verena Wolf. Computational probability for systems biology. In *Formal Methods in Systems Biology*, pages 33–47. Springer, 2008.

[SWR+11] Kevin R Sanft, Sheng Wu, Min Roh, Jin Fu, Rone Kwei Lim, and Linda R Petzold. Stochkit2: software for discrete stochastic simulation of biochemical systems with events. *Bioinformatics*, 27(17):2457–2458, 2011.

[TB04] Tianhai Tian and Kevin Burrage. Binomial leap methods for simulating stochastic chemical kinetics. *The Journal of chemical physics*, 121(21):10356–10364, 2004.

[TLS06] Bruno Tuffin, Pierre L'Écuyer, and Werner Sandmann. *Robustness properties for simulations of highly reliable systems.* Groupe d'études et de recherche en analyse des décisions, 2006.

[TWS+09] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael PH Stumpf. Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202, 2009.

[TXGB07] Tianhai Tian, Songlin Xu, Junbin Gao, and Kevin Burrage. Simulated maximum likelihood method for estimating kinetic rates in gene expression. *Bioinformatics*, 23(1):84–91, 2007.

[UAL10] Bilge Uz, Erdem Arslan, and Ian J Laurenzi. Maximum likelihood estimation of the kinetics of receptor-mediated adhesion. *Journal of theoretical biology*, 262(3):478–487, 2010.

[ULP+07] Zsolt Ugray, Leon Lasdon, John Plummer, Fred Glover, James Kelly, and Rafael Martí. Scatter search and local nlp solvers: A multistart framework for global optimization. *INFORMS Journal on Computing*, 19(3):328–340, 2007.

[VB04] Karan Vasudeva and Upinder S Bhalla. Adaptive stochastic-deterministic chemical kinetic simulations. *Bioinformatics*, 20(1):78–84, 2004.

[VK92] Nicolaas Godfried Van Kampen. *Stochastic processes in physics and chemistry*, volume 1. Elsevier, 1992.

[WGMH10] Verena Wolf, Rushil Goel, Maria Mateescu, and Thomas A Henzinger. Solving the chemical master equation using sliding windows. *BMC systems biology*, 4(1):42, 2010.

[Wil11] Darren J Wilkinson. *Stochastic modelling for systems biology*. CRC press, 2011.

[WK+10] Thomas Williams, Colin Kelley, et al. Gnuplot 4.4: an interactive plotting program. *Official gnuplot documentation, http://sourceforge. net/projects/gnuplot*, 2010.

[WLVE07] E Weinan, Di Liu, and Eric Vanden-Eijnden. Nested stochastic simulation algorithms for chemical kinetic systems with multiple time scales. *Journal of computational physics*, 221(1):158–180, 2007.

[XC08] Zhouyi Xu and Xiaodong Cai. Unbiased tau-leap methods for stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 128(15):154112, 2008.

[ZBH11] Joseph Xu Zhou, Lutz Brusch, and Sui Huang. Predicting pancreas cell fate decisions and reprogramming with a

hierarchical multi-attractor model. *PloS one*, 6(3):e14752, 2011.