INEXACT METHODS FOR THE CHEMICAL MASTER EQUATION WITH CONSTANT OR TIME-VARYING PROPENSITIES, AND APPLICATION TO PARAMETER INFERENCE

by

KHANH N. DINH

ROGER B. SIDJE, COMMITTEE CHAIR
LAYACHI HADJI
DAVID HALPERN
TATIANA MARQUEZ-LAGO
MIN SUN

A DISSERTATION

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

TUSCALOOSA, ALABAMA

2018

ABSTRACT

Complex reaction networks arise in molecular biology and many other different fields of science such as ecology and social study. A familiar approach to modeling such problems is to find their master equation. In systems biology, the equation is called the chemical master equation (CME), and solving the CME is a difficult task, because of the curse of dimensionality. The goal of this dissertation is to alleviate this curse via the use of the finite state projection (FSP), in both cases where the CME matrix is constant (if the reaction rates are time-independent) or time-varying (if the reaction rates change over time). The work includes a theoretical characterization of the FSP truncation technique by showing that it can be put in the framework of inexact Krylov methods that relax matrix-vector products and compute them expediently by trading accuracy for speed. We also examine practical applications of our work in delay CME and parameter inference through local and global optimization schemes.

DEDICATION

To my parents and other people who have cared for me like their child, and my sister, who I hope will find such joy in research as I did.

ACKNOWLEDGMENTS

My most sincere gratitude goes to my advisor, Dr. Roger Sidje, who has encouraged me through this difficult but usually exhilarating journey. He has taught me the importance of details, the art of writing scientific works and many other important skills that I need to succeed and still have to improve upon. The funding of his NSF grant DMS-1320849 was also crucial in supporting me for most of my PhD program.

I would also like to express my gratitude toward my committee members for their time and support: the external member from the University of Alabama at Birmingham School of Medicine, Dr. Tatiana Marquez-Lago, and the local members from the University of Alabama's Department of Mathematics, Dr. Layachi Hadji, Dr. David Halpern and Dr. Min Sun.

I want to thank Dr. Douglas Shepherd from the University of Colorado in Denver, who introduced me to the q-Bio Summer School, and Dr. Brian Munsky from Colorado State University in Fort Collins, who organized the school and accepted me. This summer program was one of the most important experiences in my research life, as it allowed me to meet many leaders in the field of mathematical biology, and establish ongoing collaborations with some of them. I am grateful to Dr. Marek Kimmel from Rice University in Houston, who accepted to be my mentor in the summer school, and has generously continued our project ever since.

I also want to acknowledge the generous support and encouragement from Dr. Liem Vo. I am deeply grateful to Mrs. Vo for her care and support of my well-being. Many thanks to the Department of Mathematics, the Graduate School, and the College of Arts and Sciences for funding my travels.

I want to take this occasion to thank the professors who have left lasting impacts on my mathematics education, Dr. Dang Duc Trong and Dr. Pham The Bao. I want to thank

Dr. Anthony Dang for introducing me to the University of Alabama and for his interest in my progress.

I want to acknowledge Leah Compton for helping me improve English, and David Neal, Xuan He, Alex Barnes, Sheik Ahmed, Tania Hazra, Cong Hoang, Thy Nguyen, Bryan Sandor, Douglas Weathers, Teresa Portone and Amy Puente for their friendship. Arum Lee and Tung Lam are almost like family to me. Brandon Reid, Keisha Cook and other people in our research group have been a source of new research ideas, and David Nash helped me with parallel coding, which is among the most difficult and important skills I have tried to acquire. I want to thank Timothy Homan and Summer Atkins for being great friends. Yuanyuan Song has been a true friend, and a great traveling companion. Special thanks goes to Sean Buckalew, who has supported me without asking for anything in return.

Finally, I want to thank my family for supporting me through this program. My dad was the one who showed me the important role of applied mathematics and helped me through the first steps of research. My mom is always there when I need to vent about life or work. I always enjoy spending time with Uncle Tin, whom I consider a friend. Huy Vo is my brother, with whom I have spent all my undergraduate and graduate years and I don't think our research could have been as fruitful without each other's help. My sister has given me emotional support over the years, and I wish her all the best in her research life.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A familiar approach to modeling a complex reaction network is to find the master equation that describes the joint probability density function of the population of the reactants over time. In their extensive review [4], Goutsias and Jenkinson listed many fields of science in which such an equation is of considerable importance, such as ecological networks, pharmacokinetic networks and social networks.

In systems biology, the equation is called the chemical master equation (CME) [5], and its solution is the probability distribution of finding the system in all possible states (i.e., all possible integer-valued populations of the reacting chemical species). The CME will be reviewed in detail in the next chapter.

It is easy to see why solving the CME is a formidable task: if the copy numbers of species in the system are not bounded, then there are potentially infinitely many states that the system can occupy. Even if we apply a bound on the species numbers, the size of the CME increases exponentially with the numbers of species and therefore solving for the probability distribution of all of them is numerically very expensive.

Despite this so-called "curse of dimensionality", solving the CME has been of great interest to the systems biology community, because unlike deterministic models, the CME captures the randomness of the biological processes. It has been shown in different biological contexts that single molecular events may significantly impact the process, which underlines the importance of stochasticity in these models.

The main approach to solving the CME has often been using Monte Carlo algorithms, of which Gillespie's stochastic stimulation algorithm (SSA) [6] has been extensively employed. However, the SSA can be very slow, considering the numerous runs

needed to average the probability distribution. Several modifications to improve its efficiency include the tau-leaping method [7, 8] or slow-scale SSA [9].

The finite state projection (FSP) method [10] is a different approach to provide an approximation to the actual probability distribution at the end time as well as the transient probabilities with a guaranteed accuracy. Soon after the FSP, an adaptive time-stepping version was introduced [11, 12] and followed by other variants. These algorithms have proved to give accurate probability distributions, and their speed has been helpful in gene regulation problems where a set of parameters have to be found, which can only be achieved by computing the distributions for many parameter sets and choosing the distribution closest to experimental data [13].

Classic time-stepping FSP schemes, however, cannot be employed in the basic form if the reaction rates change over time. In this case, the CME must be solved by traditional ODE solvers. There also has not been many works that review the different FSP variants in a systematic fashion, or establish the theoretical background for why they are efficient for certain biological problems. These issues will be targeted in this dissertation, which aims at providing a broad view on the theory and application of the FSP.

1.1 Outline of this dissertation

This dissertation consists of five main parts. The first part introduces the CME and surveys different variants of the FSP for solving the CME with constant rates. The second part generalizes the FSP as an inexact Krylov subspace method and discovers its analytical background. The third part examines different contexts such as delay CME that can be reformulated as a CME with time-varying rates, in which case most FSP approaches are not applicable. The fourth part introduces a new FSP approach based on the Magnus expansion for solving a CME with variable rates, and compares this algorithm with existing ODE solvers on several biological problems. The last part examines the application of the FSP in parameter inference for a stochastic biological model. Below is a breakdown of the following chapters in greater details:

- Chapter 2 sets up the chemical master equation (CME), which is the problem that will be addressed in the rest of the dissertation, as well as the finite state projection (FSP), which is the main method employed to solve the CME in this dissertation. Several variants of the adaptive time-stepping FSP in the literature are discussed.

- Chapter 3 generalizes the FSP and develops the theoretical background for the FSP truncation technique in the framework of inexact Krylov methods that relax matrix-vector products. Schemes for controlling the bounds on the residual or the error of this method are established.

- Chapter 4 reviews three different scenarios where a CME with time-varying rates arises. It may emerge naturally from the biological context where the environmental changes affect the reaction rates. On the other hand, a non-homogeneous ODE can also be expressed in the form of a CME. Finally, the CME for a biological system in which some reactions require a delay time period to finish (i.e. delay CME) can either be transformed to or approximated as a CME with time-varying rates.

- Chapter 5 reviews traditional approaches for solving a CME with variable rates: kinetic Monte Carlo methods, such as first reaction method (FRM) and stochastic simulation algorithm (SSA), and ODE solvers, such as Adams, Runge-Kutta, and Backward-differentiation formula (BDF). We then develop a new adaptive time-stepping integration algorithm for this particular problem, based on the FSP and the Magnus expansion, equipped with four different techniques for controlling the error. The chapter concludes with several numerical tests comparing the new method with existing ODE solvers.

- Chapter 6 considers an important application for the FSP methods reviewed and developed in this dissertation, which is parameter inference in a stochastic biological model through local and global optimization schemes. This is performed via a

data-driven maximum likelihood approach, where the probability distributions for many different parameter sets are computed by the FSP.

- Finally, chapter 7 summarizes the findings made in this dissertation and identifies opportunities for future work.

CHAPTER 2

UNDERSTANDING THE FINITE STATE PROJECTION

This chapter is based on work published in Physical Biology [14].

2.1 Introduction

In this chapter, we formularize the chemical master equation (CME). We also describe several variants of the finite state projection (FSP) in the literature and discuss their strengths and weaknesses.

The chapter is organized as follows: Section 6.2 describes the notation, formalizes the CME, and comments on popular Monte Carlo methods. Section 2.3 outlines the original FSP method. Section 2.4 describes a time-stepping framework, where each step consists of three stages, and we show how all variants of the FSP aimed at improving one or more of these stages. From there, the variants are presented in Sections 2.5, 2.6 and 2.7, following how they fit in the time-stepping framework. Note that each method is accompanied by a simplified pseudocode for ease of readability. Another approach of solving the CME via tensor decomposition is discussed in Section 2.8. An illustrative example is given in Section 2.9 to demonstrate how to generate the CME matrix for a specific biological case. Section 2.10 lists some available software for solving the CME by using the FSP. The strengths and weaknesses of solving the CME are discussed in Section 2.11. We finish with some concluding remarks in Section 6.6.

2.2 Chemical master equation

Consider a chemical reaction system consisting of N molecular species $S_1, \ldots, S_N$ that interact through M reactions of the form

$$R_k : \; a_{1k}S_1 + \cdots + a_{Nk}S_N \xrightarrow{c_k} b_{1k}S_1 + \cdots + b_{Nk}S_N$$

for $k = 1, \ldots, M$. The c_k are *reaction rate constants*, which are scale factors for how likely such a collision of the reactants results in a reaction. At any time, the system can be described as the numbers of copies of each species. We then can define the *state vector* of the system as the vector of these numbers:

$$\boldsymbol{x} = (x_1, \ldots, x_N)^T,$$

where x_l is a count for species S_l. We denote $\boldsymbol{x}(t)$ as the state of the system at time t.

The *propensity function* $\alpha_k(\boldsymbol{x}(t))$ of reaction R_k at the current state $\boldsymbol{x}(t)$ is defined so that the probability of such a reaction occurring during the infinitesimal time interval $[t, t + dt)$ is equal to $\alpha_k(\boldsymbol{x}(t))dt$.

When reaction R_k happens, the state vector is updated as

$$\boldsymbol{x}(t + dt) \leftarrow \boldsymbol{x}(t) + \boldsymbol{\nu}_k, \tag{2.1}$$

where the *stoichiometric vector* $\boldsymbol{\nu}_k$, representing the change in species numbers, is defined as

$$\boldsymbol{\nu}_k = (b_{1k} - a_{1k}, \ldots, b_{Nk} - a_{Nk})^T.$$

We are now interested in the probability that the system is at state $\boldsymbol{x}$ at time t, which we denote $P(\boldsymbol{x}, t) = \text{Prob}\{\boldsymbol{x}(t) = \boldsymbol{x}\}$. Assuming that we know the number of each species at $t = 0$ (from which we can deduce $P(\boldsymbol{x}, 0)$), the CME [5] dictates that

$$\frac{dP(\boldsymbol{x}, t)}{dt} = \sum_{k=1}^{M} \alpha_k(\boldsymbol{x} - \boldsymbol{\nu}_k) P(\boldsymbol{x} - \boldsymbol{\nu}_k, t) - \sum_{k=1}^{M} \alpha_k(\boldsymbol{x}) P(\boldsymbol{x}, t). \tag{2.2}$$

The expression is clear if we note that $\alpha_k(\boldsymbol{x} - \boldsymbol{\nu}_k)dt$ is the probability for state $\boldsymbol{x} - \boldsymbol{\nu}_k$ to transition to state $\boldsymbol{x}$ through reaction R_k during $[t, t + dt)$, and $\left(\sum_{k=1}^{M} \alpha_k(\boldsymbol{x})\right) dt$ is the probability for the system to escape from state $\boldsymbol{x}$ through any reaction during that same time period.

Let $\boldsymbol{X}$ be the set of all possible states, if we order these states as $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, where $\boldsymbol{x}_i = (x_{1i}, \ldots, x_{Ni})^T$ and n is the total number of states, then (2.2) defines a set of ODEs governing the change in $\boldsymbol{p}(t) = (P(\boldsymbol{x}_1, t), \ldots, P(\boldsymbol{x}_n, t))^T$:

$$\dot{\boldsymbol{p}}(t) = \boldsymbol{A} \cdot \boldsymbol{p}(t), \tag{2.3}$$

where here we set $\boldsymbol{p}(0) = (1, 0, \ldots, 0)^T$ by assuming that the system is at state $\boldsymbol{x}_1$ at $t = 0$. The transition rate matrix $\boldsymbol{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$ is defined as

$$a_{ij} = \begin{cases} -\sum_{k=1}^{M} \alpha_k(\boldsymbol{x}_j), & \text{if } i = j \\ \alpha_k(\boldsymbol{x}_j), & \text{if } \boldsymbol{x}_i = \boldsymbol{x}_j + \boldsymbol{\nu}_k \\ 0, & \text{otherwise} \end{cases} .$$

From (2.3) we can find the probability vector at the end point t_f:

$$\boldsymbol{p}(t_f) = \exp(t_f \boldsymbol{A}) \boldsymbol{p}(0),$$

where the exponential matrix is defined as

$$\exp(t_f \boldsymbol{A}) = \sum_{m=0}^{\infty} \frac{(t_f \boldsymbol{A})^m}{m!}.$$

An explicit formula for $\boldsymbol{p}(t)$ can be given only in extremely simple cases, such as monomolecular reaction systems [15], therefore it is necessary to numerically solve at a prescribed end point t_f.

The enormous size of the CME usually makes it too challenging to solve directly. The stochastic simulation algorithm (SSA) takes a piecemeal approach by computing single realizations of the state vector rather than an entire probability distribution. For each realization, the algorithm updates the state as in (2.1), by randomly choosing the

time between events, dt, and the next reaction index, k. The τ-leap variant that seeks to improve the efficiency of the SSA consists in allowing a larger time between events, $\tau \leftarrow dt$, so that more that one reaction can be accumulated in the state update.

As noted before, the cost of these methods is compounded by the multiple runs that have to be done to average the results in a Monte Carlo manner. More details about these algorithms can be found in [6, 7, 8], and further improvements in [16, 9, 17].

2.3 The original finite state projection

Unlike SSA, the FSP method seeks to directly approximate the probability density function that is the solution of (2.3). We define $\boldsymbol{X}_J$ to be a finite subset of states in $\boldsymbol{X}$, where J is the index set of those states. Consider all the other states not in $\boldsymbol{X}$ as only one state, which we call the *sink state* G. Let $\boldsymbol{A}_J$ be a submatrix of $\boldsymbol{A}$ containing only the elements on the rows and columns indexed by J, and $\boldsymbol{p}(\boldsymbol{X}_J, t)$ contains only the probabilities of states indexed in J at time t. The FSP method approximates $\boldsymbol{p}(\boldsymbol{X}, t)$ by $\boldsymbol{p}_J^{\mathrm{FSP}}(t) = \boldsymbol{p}^{\mathrm{FSP}}(\boldsymbol{X}_J, t)$, which follows the master equation

$$\begin{pmatrix} \dot{\boldsymbol{p}}_J^{\mathrm{FSP}}(t) \\ \dot{g}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_J & \boldsymbol{0} \\ -\boldsymbol{1}^T \boldsymbol{A}_J & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_J^{\mathrm{FSP}}(t) \\ g(t) \end{pmatrix}, \tag{2.4}$$

with initial distribution

$$\begin{pmatrix} \boldsymbol{p}_J^{\mathrm{FSP}}(0) \\ g(0) \end{pmatrix} = \begin{pmatrix} \boldsymbol{p}(\boldsymbol{X}_J, 0) \\ 1 - \sum \boldsymbol{p}(\boldsymbol{X}_J, 0) \end{pmatrix}.$$

The theoretical solution to this set of ODEs can be proven to be

$$\begin{cases} \boldsymbol{p}_J^{\mathrm{FSP}}(t) = \exp(t\boldsymbol{A}_J)\boldsymbol{p}_J^{\mathrm{FSP}}(0) \\ g(t) = \boldsymbol{1}^T(\boldsymbol{I} - \exp(t\boldsymbol{A}_J))\boldsymbol{P}_J^{\mathrm{FSP}}(0) + g(0) \end{cases}. \tag{2.5}$$

Note that since $\boldsymbol{A}_J$ is extracted from $\boldsymbol{A}$, $\boldsymbol{p}_J^{\mathrm{FSP}}(t)$ has a different statistical meaning from $\boldsymbol{p}(\boldsymbol{X}_J, t)$: the elements in $\boldsymbol{p}_J^{\mathrm{FSP}}(t)$ are the probabilities that the system is at the corresponding states at t and has never left $\boldsymbol{X}_J$ during $[0, t)$. On the other hand, $g(t)$ is the probability that the system visited the sink state G at least once during $[0, t)$ (also note that if $\boldsymbol{X}_J$ contains the initial state then $g(0) = 0$). In [10] the authors showed that these facts imply

$$\boldsymbol{p}(\boldsymbol{X}_J, t) \geq \boldsymbol{p}_{J_2}^{\mathrm{FSP}}(t) \geq \boldsymbol{p}_{J_1}^{FSP}(t) \geq \boldsymbol{0} \tag{2.6}$$

if $J_1 \subset J_2$, and

$$\left\| \begin{pmatrix} \boldsymbol{p}(\boldsymbol{X}_J, t) \\ \boldsymbol{p}(\boldsymbol{X}_{J'}, t) \end{pmatrix} - \begin{pmatrix} \boldsymbol{p}_J^{\mathrm{FSP}}(t) \\ \boldsymbol{0} \end{pmatrix} \right\|_1 = g(t) \tag{2.7}$$

where J' is the index set of the states not in J. The significance of these two properties cannot be overstated: (2.6) guarantees that the FSP approximation only improves monotonically element-wise if we keep expanding the state space, and (2.7) gives us the exact evaluation of the 1-norm error.

The original FSP algorithm follows directly from the solution (2.5) and these two observations.

Original FSP algorithm
(B. Munsky, M. Khammash, [10], 2006)

Find the FSP approximation with 1-norm error less than ϵ at t_f:

1: Initialize $i = 0$, J_0 and $\boldsymbol{A}_{J_0}$.

2: Approximate $\exp(t_f \boldsymbol{A}_{J_i}) \boldsymbol{p}(\boldsymbol{X}_{J_i}, 0)$.
If $\boldsymbol{1}^T \exp(t_f \boldsymbol{A}_{J_i}) \boldsymbol{p}(\boldsymbol{X}_{J_i}, 0) \geq 1 - \epsilon$, then stop. We have the approximation

$$\boldsymbol{p}(\boldsymbol{X}_{J_i}, t_f) \approx \exp(t_f \boldsymbol{A}_{J_i}) \boldsymbol{p}(\boldsymbol{X}_{J_i}, 0),$$

with

$$\| \boldsymbol{p}(\boldsymbol{X}_{J_i}, t_f) - \exp(t_f \boldsymbol{A}_{J_i}) \boldsymbol{p}(\boldsymbol{X}_{J_i}, 0) \|_1 \leq \epsilon.$$

3: Increment i and add more states into J_i, then return to step 2.

2.4 Time-stepping FSP variations

There has been great interest in modifying the FSP algorithm into a time-stepping scheme. Many approaches that have been developed for this purpose follow the same framework demonstrated in the following variable time-stepping FSP algorithm.

The algorithm divides $[0, t_f]$ into small intervals

$$0 = t_0 < t_1 < \cdots < t_{K+1} = t_f.$$

At every interval $[t_k, t_{k+1}]$, the quest is then to find the index set J_k of the most likely states over that interval. The probability vector at t_{k+1} is then calculated by

$$\boldsymbol{p}(t_{k+1}) \approx \exp(\tau_k \boldsymbol{A}_{J_k}) \boldsymbol{p}^{\mathrm{FSP}}_{J_k}(t_k), \tag{2.8}$$

and the algorithm moves on to the next time interval.

Though not always the case, the problem of approximating (2.8) can be even further reduced before solving, especially when special properties of the reactions or the species are realized. In these cases, the reduced system will be preconditioned in step 2 of the algorithm. Apart from that, steps 1 and 3 are taken from the original FSP algorithm. Note that our notation for the FSP approximation is $\boldsymbol{p}_{J_k}(t_k) = \boldsymbol{p}^{\mathrm{FSP}}_{J_k}(t_k) = \boldsymbol{p}^{\mathrm{FSP}}(\boldsymbol{X}_{J_k}, t_k)$ and not to be confused with the true solution $\boldsymbol{p}(\boldsymbol{X}_{J_k}, t_k)$.

Variable time-stepping FSP algorithm
(K. Burrage, M. Hegland, S. MacNamara, R. Sidje, [11], 2006.
B. Munsky, M. Khammash, [12], 2007.)

0: Start from $k = 0$, $t_k = t_0$.
1: Find the time step τ_k, and the state space $\boldsymbol{X}_{J_{k+1}}$ containing states most likely over $[t_k, t_{k+1}]$ where $t_{k+1} = t_k + \tau_k$.
2: Precondition the reduced system before solving.
3: Approximate

$$\boldsymbol{p}(t_{k+1}) \approx \exp(\tau_k \boldsymbol{A}_{J_{k+1}}) \boldsymbol{p}_{J_k}(t_k).$$

4: If $t_{k+1} < t_f$, set $k = k + 1$ and go to Step 1.

Note that the variable time-stepping FSP algorithm serves more as a framework, because each step from 1 to 3 can be modified for the specific biological problem. The next three sections will drill further into these steps: section 2.5 considers different strategies to update the state space, section 2.6 discusses some methods to precondition the FSP, and section 2.7 introduces some techniques to compute the action of matrix exponential on a vector while taking advantage of the fact that the entire matrix exponential is not needed.

2.5 Update the state space

It is clear that a crucial part of the variable time-stepping FSP algorithm is to find a strategy for updating the state space: it needs to contain enough states that contribute the most to the probability mass, but a too large state space results in a bigger $\boldsymbol{A}_J$ and a more time-consuming evaluation of its matrix exponential.

2.5.1 Update by r-step reachability

The original paper by Munsky and Khammash [10] has a suggestion for how to expand the state space through the concept of reachability. They made an observation that, the system ends up in a state at t_{k+1} by jumping from a state in t_k through only finitely many states in $\boldsymbol{X}$. Therefore, much of the probability mass at t_{k+1} will be contained in the set of the states either in $\boldsymbol{X}_{J_{k-1}}$ or can be reached from a state in $\boldsymbol{X}_{J_{k-1}}$ within one reaction, we call the index set of these states $\mathcal{R}_1(J_{k-1})$:

$$\boldsymbol{X}_{\mathcal{R}_1(J_{k-1})} = \boldsymbol{X}_{J_{k-1}} \bigcup_{i=1}^{M} \{\boldsymbol{x} + \boldsymbol{\nu}_i \in \boldsymbol{X}; \boldsymbol{x} \in \boldsymbol{X}_{J_{k-1}}\}.$$

Inductively, the index set of states either in $\boldsymbol{X}_{J_{k-1}}$ or within r reactions from $\boldsymbol{X}_{J_{k-1}}$ can be defined as $\mathcal{R}_r(J_{k-1}) = \mathcal{R}_1(\mathcal{R}_{r-1}(J_{k-1}))$.

2.5.2 Update by multiple absorbing states

One disadvantage of the r-step reachability algorithm is that of the states that can be reached from the current state space, the probabilities of many are so low that including

Update by 2.5.1 r-step reachability
(B. Munsky, M. Khammash, [10], 2006)

0: $J_k = J_{k-1}$
1: Repeat r times:

$$J_k = \mathcal{R}_1(J_k).$$

them in the state space only results in a big matrix without a noticeable improvement in error. The multiple absorbing states method, discussed in [12], attempts to solve the problem. Instead of grouping all the states not in $\boldsymbol{X}_{J_k}$ into only one sink state, the method divides them into L sink states $G_1, \ldots, G_L$ such that G_i contains all states that escape $\boldsymbol{X}_{J_k}$ through reaction R_i. Letting $g_i(t)$ be the probability that the system escapes to G_i from $\boldsymbol{X}_{J_k}$, and $\boldsymbol{g}(t) = (g_1(t), \ldots, g_L(t))^T$, the system then follows the ODEs

$$\begin{pmatrix} \dot{\boldsymbol{p}}_{J_k}(t) \\ \dot{\boldsymbol{g}}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_{J_k} & \boldsymbol{0} \\ \boldsymbol{Q} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_{J_k}(t) \\ \boldsymbol{g}(t) \end{pmatrix} \tag{2.9}$$

instead of (2.4), where

$$\boldsymbol{Q}(i,j) = \begin{cases} \alpha_i(\boldsymbol{x}_j), & \text{if } (\boldsymbol{x}_j + \boldsymbol{\nu}_i) \notin \boldsymbol{X}_{J_k} \\ 0, & \text{otherwise} \end{cases}.$$

The solution of (2.9) at t_{k+1} is

$$\begin{cases} \boldsymbol{p}_{J_k}(t_{k+1}) = \exp(\tau_k \boldsymbol{A}_{J_k}) \boldsymbol{p}_{J_k}(t_k) \\ \boldsymbol{g}(t_{k+1}) = \left(\int_0^\tau \boldsymbol{Q} \exp(\xi \boldsymbol{A}_{J_k}) d\xi \right) \boldsymbol{p}_{J_k}(t_k) + \boldsymbol{g}(t_k) \end{cases}. \tag{2.10}$$

As we can see, the set of ODEs (2.9) does not give us a better approximation for $\boldsymbol{p}_{J_k}(t)$, but it gives us information about what reactions leak the most probability mass from $\boldsymbol{X}_{J_k}$. To expand the state space, we then only consider the directions of the reactions contributing much to the probability mass drop. In [18], the absorbing states are defined

by an arbitrary set of nonlinear inequalities, which can avoid the stiffness of the matrix $\boldsymbol{A}_{J_k}$ and therefore can be much more efficient.

Update by 2.5.2 r-step reachability and multiple absorbing states
(B. Munsky, M. Khammash, [12], 2007, [18], 2011)

0: Start from $k = 0$, time-step τ_k and tolerance on error ϵ_k.
1: Define a set of functions $\{f_1, \ldots, f_L\}$ and a set of bounds $\{b_1, \ldots, b_L\}$.
2: $\boldsymbol{X}_{J_k}$ is defined as all states $\boldsymbol{x}$ so that

$$f_i(\boldsymbol{x}) \leq b_i, \quad i = 1, \ldots, L,$$

and G_i contains the all states exiting $\boldsymbol{X}_{J_k}$ through inequality f_i.
Note: The next two stages in the Variable time-stepping FSP algorithm solve for $\boldsymbol{p}(t_{k+1})$ and $\boldsymbol{g}(t_{k+1})$ by applying (2.10). In case $\mathbf{1}^T \boldsymbol{g}(t_{k+1}) > \epsilon_k$, we can redo the process and choose a larger b_i when $g_i(t_{k+1})$ is big.

2.5.3 Update by sliding windows

The sliding windows algorithm updates the state space from $\boldsymbol{X}_{J_{k-1}}$ at t_k to $\boldsymbol{X}_{J_k}$ at t_{k+1} by constructing a window $\boldsymbol{W}_k \subset \boldsymbol{X}$ that aims to contain the most probability mass during $[t_k, t_{k+1}]$. We then can solve for the probability distribution of $\boldsymbol{W}_k$ at t_{k+1}, but the whole vector is not stored. Only the states in $\boldsymbol{W}_k$ having a considerable probability at t_{k+1} are kept in $\boldsymbol{X}_{J_k}$. Therefore the algorithm can avoid having a big state space, and as a result, computing the probability vector for the next time step can be more efficient.

$\boldsymbol{W}_k$ is constructed by realizing which states are frequently visited by the system during $[t_k, t_{k+1}]$. The task is done by applying a stochastic approach to estimate the largest and smallest populations attained by each species over $[t_k, t_{k+1}]$, and then use these extremes to form the boundaries of the window $\boldsymbol{W}_k$.

Since the exact transient dynamics during $[t_k, t_{k+1}]$ is not needed, the extremes are found by a crude random approximation during this time period, instead of using the SSA which could be slow. The time interval $[t_k, t_{k+1}]$ is divided into small equal intervals

$$[t_k, t_k + \Delta, t_k + 2\Delta, \ldots, t_{k+1}].$$

During each interval $[t^{(l)}, t^{(l)} + \Delta]$, the propensity of each reaction R_i is assumed to remain constant to $\alpha_i(\boldsymbol{x}^{(l)})$ where $\boldsymbol{x}^{(l)}$ is the state at $t^{(l)}$. This means that the number of reactions R_i to occur during $[t^{(l)}, t^{(l)} + \Delta]$ is Poisson distributed, with parameter $\alpha_i(\boldsymbol{x}^{(l)})\Delta$. Taking into account that the standard deviation of the Poisson distribution is $\sqrt{\alpha_i(\boldsymbol{x}^{(l)})\Delta}$, statistically it can be assumed that the actual number of reactions R_i happening during $[t^{(l)}, t^{(l)} + \Delta]$ is at most

$$\kappa_i^+(\boldsymbol{x}^{(l)}, \Delta) = \alpha_i(\boldsymbol{x}^{(l)})\Delta + \sqrt{\alpha_i(\boldsymbol{x}^{(l)})\Delta}$$

and at least

$$\kappa_i^-(\boldsymbol{x}^{(l)}, \Delta) = \max\left(0, \alpha_i(\boldsymbol{x}^{(l)})\Delta - \sqrt{\alpha_i(\boldsymbol{x}^{(l)})\Delta}\right).$$

Because the interest is in building a window containing considerable probability mass, either of these two extremes is assumed to have happened, for each reaction of type R_i. Continuing until reaching t_{k+1}, we have one trajectory where the worst case scenario happens at each step. The maximum and minimum of x_d, the number of the dth species of state $\boldsymbol{x}$, along all trajectories are then the boundaries for the window $\boldsymbol{W}_k$.

Update by 2.5.3 Sliding windows
(V. Wolf, R. Goel, et al., [19], 2010)

0: Start with $\boldsymbol{X}_{J_{k-1}}$ at t_k, time-step τ_k, parameter δ.
1: Find the extremes of each dimension d for the window, b_d^+ and b_d^-. These are estimates of the largest and smallest populations attained by the dth species over $[t_k, t_{k+1}]$.
2: The state window will be $\boldsymbol{W}_k = \boldsymbol{X}_{J_{k-1}} \cup \{\boldsymbol{x} \in \boldsymbol{X} : b_d^- \leq x_d \leq b_d^+\}$.
3: Solve for
$$\boldsymbol{p}(\boldsymbol{W}_k, t_{k+1}) = \exp(\tau_k \boldsymbol{A}_{\boldsymbol{W}_k})\boldsymbol{p}(\boldsymbol{W}_k, t_k).$$
4: $\boldsymbol{X}_{J_k} = \{\boldsymbol{x} \in \boldsymbol{W}_k : P(\boldsymbol{x}, t_{k+1}) > \delta\}$, and $\boldsymbol{p}(\boldsymbol{X}_{J_k}, t_{k+1})$ is truncated accordingly from $\boldsymbol{p}(\boldsymbol{W}_k, t_{k+1})$.

2.5.4 Update by the optimal state space

The "optimal" FSP method [20] underlines the problem that methods such as r-step reachability has and that the sliding windows tries to solve: expanding the state space

without removing any or most improbable states can result in an unnecessarily big problem to solve. The proposal of the Optimal FSP method is intuitive: after expanding the state space from $\boldsymbol{X}_{J_{k-1}}$ at time t_k to $\boldsymbol{X}_{J_k}$ at time t_{k+1} using conventional methods, we solve for $\boldsymbol{p}(t_{k+1})$ and then remove the states in $\boldsymbol{X}_{J_k}$ whose probabilities at t_{k+1} are too small.

The question that arises is then how many states do we remove at each step. The algorithm proposes the following approach, consisting of two steps:

- Step 1: Find the state space $J_k^{\epsilon/2}$ by any FSP method (the work applied r-step reachability) so that the 1-norm error at t_{k+1} is less than a prescribed $\frac{\epsilon}{2}$.
- Step 2: Find and delete the states in $J_k^{\epsilon/2}$ with the smallest probabilities at t_{k+1} that add up to $\frac{\epsilon}{2}$, resulting in the state space J_k.

J_k is then guaranteed to have the 1-norm error at t_{k+1} of at most ϵ.

We need to point out that the method is only "optimal" in the sense that the resulting state space has the fewest elements while still ensuring that the the 1-norm error is less than a prescribed ϵ.

However, the fact that the first step in the method applies r-step reachability with error $\epsilon/2$ instead of ϵ as in the original r-step reachability ensures that finding the state space will be much slower than other methods, although approximating $\boldsymbol{p}_k(t_{k+1})$ is faster.

2.5.5 Update by GORDE

The Gated One Reaction Domain Expansion (GORDE) is another method targeted at making r-step reachability more efficient. However, instead of optimizing the state space after solving for the probability vector at t_{k+1} like the Optimal FSP method, GORDE estimates the probabilities of the likely reachable states using a *gating function*.

The main disadvantage of r-step reachability is that it does not evaluate the likelihood of the new states when expanding the state space. The result is that the algorithm expands in the directions of the unlikely states as much as in the directions of the more probable ones. GORDE seeks to solve this by assigning every state $\boldsymbol{x}$ that is m

Update by 2.5.4 Optimal FSP method
(V. Sunkara, M. Hegland, [20], 2012)

0: Start from $\boldsymbol{X}_{J_{k-1}}$ at t_k, time-step τ_k and tolerence ϵ_k.
1: Apply r-step reachability to find J_k so that

$$1 - \mathbf{1}^T \boldsymbol{p}(t_{k+1}) < \frac{\epsilon_k}{2}.$$

2: Sort $\boldsymbol{p}_k(t_{k+1})$ in descending order, then remove the states with the smallest probabilities at t_{k+1} into J' so that

$$\mathbf{1}^T \boldsymbol{p}(\boldsymbol{X}_{J'}, t_{k+1})$$

is as close to $\frac{\epsilon_k}{2}$ as possible.
3: Compress the state space at t_{k+1}:

$$J_k \leftarrow J_k - J'.$$

J_k is the smallest state index set so that

$$1 - \mathbf{1}^T \boldsymbol{p}_k(t_{k+1}) < \epsilon_k.$$

steps from $\boldsymbol{X}_{J_{k-1}}$ with the *gating value* $u_m(\boldsymbol{x})$, the probability that the system travels from $\boldsymbol{X}_{J_{k-1}}$ to $\boldsymbol{x}$ in exactly m reactions during $[t_k, t_{k+1}]$ and has stayed there since. $u_m(\boldsymbol{x})$ is therefore an upper bound of and a crude approximation for $P(\boldsymbol{x}, t_{k+1})$. It acts like a weight function integrated in r-step reachability: the algorithm only expands in the directions of states $\boldsymbol{x}$ with larger $u_m(\boldsymbol{x})$.

The most practical fact about the gating function is that it can be calculated inductively in m:

$$u_m(\boldsymbol{x}) = \sum_{i=1}^{M} \frac{\alpha_i(\boldsymbol{x} - \boldsymbol{\nu}_i)}{\alpha_{\text{sum}}(\boldsymbol{x} - \boldsymbol{\nu}_i)} \times (1 - e^{-\alpha_{\text{sum}}(\boldsymbol{x} - \boldsymbol{\nu}_i)\tau_k}) u_{m-1}(\boldsymbol{x} - \boldsymbol{\nu}_i). \tag{2.11}$$

The algorithm for GORDE is then as follows: the gating values for states in $\boldsymbol{X}_{J_{k-1}}$ are initialized as $u_0(\boldsymbol{X}_{J_{k-1}}) = \boldsymbol{p}_{k-1}(t_k)$. $\boldsymbol{X}_{J_{k-1}}$ is then expanded in 1-step into $\widetilde{\boldsymbol{\nabla}}_1$, and the gating values for $\widetilde{\boldsymbol{\nabla}}_1$ are then evaluated by (2.11). If the gating values for the entire $\widetilde{\boldsymbol{\nabla}}_1$ sum up to be less than a prescribed tolerence ϵ_k, then the algorithm stops and $\boldsymbol{X}_{J_k} = \boldsymbol{X}_{J_{k-1}} \cup \widetilde{\boldsymbol{\nabla}}_1$. If not, choose the smallest set $\boldsymbol{\nabla}_1 \subset \widetilde{\boldsymbol{\nabla}}_1$ containing the highest gating

values and expand only from that set in 1-step into $\widetilde{\boldsymbol{\nabla}}_2$, with the new tolerence equals ϵ_k substracted by the gating values of states in $\widetilde{\boldsymbol{\nabla}}_1 - \boldsymbol{\nabla}_1$ and redo the whole process. In the end, if the gating values of all states in $\widetilde{\boldsymbol{\nabla}}_m$ add up to less than the tolerence, then the state space at t_{k+1} is

$$\boldsymbol{X}_{J_k} = \boldsymbol{\nabla}_0 \cup \cdots \cup \boldsymbol{\nabla}_{m-1} \cup \widetilde{\boldsymbol{\nabla}}_m.$$

Update by 2.5.5 GORDE
(V. Sunkara, [21], 2013)

0: Start from $\boldsymbol{X}_{J_{k-1}}$ at t_k, time-step τ_k and tolerence ϵ_k.
1: Initialize $\boldsymbol{\nabla}_0 = \boldsymbol{X}_{J_{k-1}}$, $u_0(\boldsymbol{x}) = P(\boldsymbol{x}, t_k)$ and $\tau_0 = \epsilon_k$.
2: For $m = 1, 2, \ldots$

- Expand $\boldsymbol{\nabla}_{m-1}$ by 1-step reachability:

$$\widetilde{\boldsymbol{\nabla}}_m \leftarrow \mathcal{R}_1(\boldsymbol{\nabla}_{m-1}).$$

- Compute the gating function for $\boldsymbol{x} \in \widetilde{\boldsymbol{\nabla}}_m$:

$$u_m(\boldsymbol{x}) = \sum_{i=1}^{M} \frac{\alpha_i(\boldsymbol{x} - \boldsymbol{\nu}_i)}{\alpha_{\text{sum}}(\boldsymbol{x} - \boldsymbol{\nu}_i)} \times (1 - e^{-\alpha_{\text{sum}}(\boldsymbol{x} - \boldsymbol{\nu}_i)\tau_k}) u_{m-1}(\boldsymbol{x} - \boldsymbol{\nu}_i),$$

  with $\alpha_{\text{sum}} = \sum_{i=1}^{M} \alpha_i$.

- Compress $\widetilde{\boldsymbol{\nabla}}_m$ into the smallest $\boldsymbol{\nabla}_m$ so that

$$\sum_{\boldsymbol{x} \in \widetilde{\boldsymbol{\nabla}}_m - \boldsymbol{\nabla}_m} u_m(\boldsymbol{x}) < \tau_{m-1}.$$

- Stop the loop if $|\boldsymbol{\nabla}_m| = 0$.
  Otherwise update

$$\tau_m = \tau_{m-1} - \sum_{\boldsymbol{x} \in \widetilde{\boldsymbol{\nabla}}_m - \boldsymbol{\nabla}_m} u_m(\boldsymbol{x}).$$

3: The next state space at t_{k+1} is

$$\boldsymbol{X}_{J_k} = \boldsymbol{\nabla}_0 \cup \cdots \cup \boldsymbol{\nabla}_{m-1} \cup \widetilde{\boldsymbol{\nabla}}_m.$$

2.5.6 Update by SSA

The SSA-driven method [1] is a combination of the r-step reachability and an approach similar to sliding windows. However, instead of forming the boundaries of a hyper-rectangle as in the sliding windows method, it uses SSA trajectories to build a collection of sets that can possibly be disjoint.

At every step, the method eliminates the states in $\boldsymbol{X}_{J_{k-1}}$ that have become improbable. It does so by applying the condition

$$J_{k-\frac{2}{3}} = \{i \in J_{k-1} : \mu(\boldsymbol{x}_i) \geq \epsilon\},$$

where $\mu(\boldsymbol{x}_i)$ is a dropping criterion, e.g., $\mu(\boldsymbol{x}_i) = \max_{j=k-\ell,\ldots,k} p_i(t_j)$ calculates the highest probability $\boldsymbol{x}_i$ has in the last ℓ steps.

The method then runs the SSA from the states in $J_{k-\frac{2}{3}}$ and saves all the states visited along the trajectories as $J_{k-\frac{1}{3}}$. To smooth these random trajectories the method further applies r-step reachability on $J_{k-\frac{1}{3}}$. The result of this will be the state space J_k at t_{k+1}.

A central feature to the efficiency of the method is its "lazy evaluation". The SSA runs are only performed when deemed necessary by an error control mechanism, and even then, the SSA trajectories are only extended as far as needed for the suitability of J_k.

Update by 2.5.6 SSA-driven method
(R. Sidje, H. Vo, [1], 2015)

0: Start from $\boldsymbol{X}_{J_{k-1}}$ at t_k, time-step τ_k, parameters ℓ and r, and the tolerence ϵ.
1: Eliminate the states in $\boldsymbol{X}_{J_{k-1}}$ with low probabilities: $J_{k-\frac{2}{3}} = \{i \in J_{k-1} : \mu(\boldsymbol{x}_i) \geq \epsilon\}$, where $\mu(\boldsymbol{x}_i) = \max_{j=k-\ell,\ldots,k} p_i(t_j)$.
2: $J_{k-\frac{1}{3}} = \bigcup_{i \in J_{k-\frac{2}{3}}} \mathrm{SSA}(\boldsymbol{x}_i, t_k, \tau_k)$.
3: J_k is the r-step expansion of $J_{k-\frac{1}{3}}$.

2.6 Precondition the FSP

The original FSP algorithm proceeds with the matrix exponential right after finding the state space. This can sometimes be slow, either because there is a difference in the magnitudes of the reaction rate constants, which causes the matrix to be stiff, or the state space is simply too large. Time scale separation [22] improves the numerical performance in the former case, by applying the perturbation theory, and aggregation [23] solves the latter case by simply grouping states together.

2.6.1 Precondition by time scale separation

Time scale separation is based on the authors' observation in [22] that in some biological cases, some reactions can have higher propensities and therefore happen more frequently than other reactions. The result is that there are clusters of states, the states within the same cluster can transition regularly to each other, and transitions between clusters are rare, which implies that the generator $\boldsymbol{A}$ can be divided into

$$\boldsymbol{A} = \boldsymbol{H} + \epsilon \boldsymbol{V},$$

where $\boldsymbol{H}$ is a block diagonal matrix, and $\epsilon \ll 1$. Each block of $\boldsymbol{H}$ represents a proper master equation for the states in one cluster, and $\epsilon \boldsymbol{V}$ contains the transition rates between these clusters.

Since the blocks of $\boldsymbol{H}$ represent ODEs for a proper master equation, and their dimensions are much less than the dimension of $\boldsymbol{A}$, it is computationally cheap to compute their eigensystems, which is equivalent to having the eigensystem of $\boldsymbol{H}$:

$$\boldsymbol{S}: \quad \boldsymbol{S}^{-1}\boldsymbol{H}\boldsymbol{S} = \tilde{\boldsymbol{\Lambda}}$$

$\boldsymbol{S}$ has same block diagonal structure as $\boldsymbol{H}$

$\tilde{\boldsymbol{\Lambda}} = diag(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n)$

We rearrange $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n$ into a decreasing sequence

$$Re(\lambda_1) \geq \cdots \geq Re(\lambda_n).$$

Notice that if $\boldsymbol{H}$ has m blocks, then $\lambda_1 = \cdots = \lambda_m = 0$, Let $\boldsymbol{S}^R \in \mathbb{R}^{N \times m}$ and $\boldsymbol{S}^L \in \mathbb{R}^{m \times N}$ contain the right and left eigenvectors of $\boldsymbol{H}$ for these zero eigenvalues (which construct the right and left null-spaces of $\boldsymbol{H}$), respectively. Note that the left eigenvectors are all $\mathbf{1}$.

Using perturbation theory, the authors observed that the projection

$$\begin{aligned} \widetilde{\boldsymbol{V}} &= \boldsymbol{S}^L \boldsymbol{V} \boldsymbol{S}^R \\ \boldsymbol{\mathcal{V}}(t) &= \boldsymbol{S}^R \exp(\epsilon \widetilde{\boldsymbol{V}} t) \boldsymbol{S}^L \end{aligned} \tag{2.12}$$

gives the asymptotic approximation to the problem:

$$t > T(\epsilon): \quad \|\boldsymbol{p}(t) - \boldsymbol{\mathcal{V}}(t)\boldsymbol{p}(0)\| = \mathcal{O}(\epsilon). \tag{2.13}$$

The time it takes for the approximation to be applicable is estimated to be

$$T(\epsilon) \sim ln(\epsilon / Re(\lambda_{m+1})).$$

If the distinction between the clusters of states is clear, then it is guaranteed that $Re(\lambda_{i \geq m}) \ll -\epsilon$, and $T(\epsilon)$ is small. However, there can be cases where the λ_i's decrease only mildly at first, and $T(\epsilon)$ can be larger than the end point t_f. When such a problem is encountered, we can simply add in the right and left eigenvector of $\boldsymbol{H}$ corresponding to λ_{m+1} in $\boldsymbol{S}^L$ and $\boldsymbol{S}^R$ (the left eigenvector no longer being $\mathbf{1}$). The projection (2.12) then satisfies (2.13) with

$$T(\epsilon) \sim ln(\epsilon / Re(\lambda_{m+2})).$$

If the new time restriction is still too big, we can keep adding in eigenvectors of $\boldsymbol{H}$ until

the condition $T(\epsilon) < \tau$ is satisfied.

Precondition by 2.6.1 Time scale separation
(S. Peles, B. Munsky, M. Khammash, [22], 2006)

0: Start with $\boldsymbol{A}_{J_k}$ and time-step τ_k. Separate $\boldsymbol{A}_{J_k}$ into $\boldsymbol{H}$ and $\epsilon\boldsymbol{V}$, and m is the number of blocks in $\boldsymbol{H}$.
1: Find $\boldsymbol{S}^R$, $\boldsymbol{S}^L$ corresponding to $\lambda_1, \ldots, \lambda_m$.
2: $T(\epsilon) = ln(\epsilon / Re(\lambda_{m+1}))$. Continue if $T(\epsilon) < \tau_k$, otherwise increase m and return to Step 1.
3: Compute $\widetilde{\boldsymbol{V}} = \boldsymbol{S}^L \boldsymbol{V} \boldsymbol{S}^R$.
4: Approximate $\exp(\epsilon\tau_k \widetilde{\boldsymbol{V}})$.
5: Compute $\boldsymbol{\mathcal{V}}(\tau_k) = \boldsymbol{S}^R \exp(\epsilon\tau_k \widetilde{\boldsymbol{V}}) \boldsymbol{S}^L$, then we have

$$\boldsymbol{p}_k(t_{k+1}) \approx \boldsymbol{\mathcal{V}}(\tau_k) \boldsymbol{p}_{k-1}(t_k).$$

2.6.2 Precondition by aggregation

There are many cases where the FSP matrix $\boldsymbol{A}_{J_k}$, already reduced from $\boldsymbol{A}$, is still too large to store or compute. One method to even further reduce the size of the matrix is aggregation [23].

The method partitions J_k into a small number of disjoint subsets

$$J_k = \bigcup_{\ell} J_{k,\ell}.$$

Let $\boldsymbol{y}_\ell$ be one state representing the whole set $\boldsymbol{X}_{J_{k,\ell}}$, and its probability equals the probability mass of $\boldsymbol{X}_{J_{k,\ell}}$:

$$P(\boldsymbol{y}_\ell, t) = \sum_{\boldsymbol{x} \in \boldsymbol{X}_{J_{k,\ell}}} P(\boldsymbol{x}, t).$$

Let the *aggregation operator* $\boldsymbol{E}$ define this conversion from the probability vector of $\boldsymbol{X}_{J_{k,\ell}}$ into the probability distribution of $\boldsymbol{Y} = \{\boldsymbol{y}_\ell\}$:

$$\boldsymbol{p}(\boldsymbol{Y}, t) = \boldsymbol{E} \cdot \boldsymbol{p}(\boldsymbol{X}_{J_k}, t),$$

and the *disaggregation operator* $\boldsymbol{F}$, which approximates $\boldsymbol{p}(\boldsymbol{X}_{J_{k,\ell}}, t)$ from $\boldsymbol{p}(\boldsymbol{Y}, t)$, is the

right inverse of $\boldsymbol{E}$:

$$\boldsymbol{E} \cdot \boldsymbol{F} = \boldsymbol{I}_{\boldsymbol{Y}}.$$

The Markov generator for $\boldsymbol{Y}$ can be reduced from $\boldsymbol{A}_{J_k}$:

$$\boldsymbol{B} = \boldsymbol{E} \cdot \boldsymbol{A}_{J_k} \cdot \boldsymbol{F}.$$

It can be shown that $\boldsymbol{B}$ represents a proper master equation: its off-diagonal elements are nonnegative and each column sums up to 0. The ODEs for the distribution of $\boldsymbol{Y}$ are then

$$\begin{cases} \dot{\boldsymbol{p}}(\boldsymbol{Y}, t) = \boldsymbol{B} \cdot \boldsymbol{p}(\boldsymbol{Y}, t), \quad t > t_k \\ \boldsymbol{p}(\boldsymbol{Y}, t_k) = \boldsymbol{E} \cdot \boldsymbol{p}(\boldsymbol{X}_{J_k}, t_k) \end{cases}$$

which we can solve using techniques in Section 2.7, then $\boldsymbol{p}(\boldsymbol{X}_{J_{k,\ell}}, t_{k+1})$ can be deduced by

$$\boldsymbol{p}(\boldsymbol{X}_{J_k}, t_{k+1}) = \boldsymbol{F} \cdot \boldsymbol{p}(\boldsymbol{Y}, t_{k+1}), \tag{2.14}$$

which means that the states in each $\boldsymbol{X}_{J_{k,\ell}}$ will have the same probability. In practice, if the shape of the probability distribution is known, then different strategies to disaggregate from $\boldsymbol{p}(\boldsymbol{Y}, t)$ to $\boldsymbol{p}(\boldsymbol{X}_{J_k}, t)$ can be employed instead of (2.14), most notably interpolation. We refer to [24] for different methods toward this end.

2.7 Approximate the solution

The last stage in each step of the adaptive time-stepping FSP algorithm is to approximate $\boldsymbol{p}_k(t_{k+1}) = \exp(\tau_k \boldsymbol{A}_{J_k}) \cdot \boldsymbol{p}_{k-1}(t_k)$. Obviously, $\boldsymbol{p}_k$ is the solution of

$$\dot{\boldsymbol{p}}(t) = \boldsymbol{A}_{J_k} \cdot \boldsymbol{p}(t) \tag{2.15}$$

at t_{k+1}, where $\boldsymbol{p}(t_k) = \boldsymbol{p}_{k-1}(t_k)$. Therefore, standard ODE solution techniques can be applied, for instance Runge-Kutta methods.

Precondition by 2.6.2 Aggregation
(M. Hegland, C. Burden, L. Santoso, S. MacNamara, H. Booth, [23], 2005)

0: Start with $\boldsymbol{X}_{J_k}$ and $\boldsymbol{A}_{J_k}$ and τ_k.
1: Divide $\boldsymbol{X}_{J_k}$ into $\boldsymbol{X}_{J_{k,\ell}}$'s.
2: Find the operators $\boldsymbol{E}$ and $\boldsymbol{F}$ based on the aggregation.
3: Compute

$$\boldsymbol{B} = \boldsymbol{E} \cdot \boldsymbol{A}_{J_k} \cdot \boldsymbol{F},$$

and

$$\boldsymbol{p}(\boldsymbol{Y}, t_k) = \boldsymbol{E} \cdot \boldsymbol{p}_k(t_k).$$

4: Solve for

$$\boldsymbol{p}(\boldsymbol{Y}, t_{k+1}) = \exp(\tau_k \boldsymbol{B}) \boldsymbol{p}(\boldsymbol{Y}, t_k).$$

5: We have

$$\boldsymbol{p}_k(t_{k+1}) \approx \boldsymbol{F} \cdot \boldsymbol{p}(\boldsymbol{Y}, t_{k+1}),$$

or by other methods, as reported in [24].

Here we will address some techniques to solve (2.15) which take advantage of the fact that $\boldsymbol{A}_{J_k}$ is constant over time. Krylov subspace techniques have proved very efficient for solving this kind of problem for large sparse $\boldsymbol{A}_{J_k}$, by projecting it down to a small dense matrix, the exponential of which can then be approximated by Padé or Chebyshev polynomials.

A different approach to solving this problem is the uniformization method, which truncates the Taylor series expansion of the matrix.

2.7.1 Approximate by uniformization

Even if $\boldsymbol{A}_{J_k}$ is preconditioned or not, each step in the variable time-stepping FSP algorithm ends with approximating the probability vector at time t_{k+1}, in the form of a matrix exponential multiplied by a vector:

$$\boldsymbol{p}_{J_k}(t_{k+1}) \approx \exp(\tau_k \boldsymbol{A}_{J_k}) \boldsymbol{p}_{J_{k-1}}(t_k).$$

We will simplify the notation so that the problem is approximating

$$\exp(\tau \boldsymbol{A})\boldsymbol{v}, \tag{2.16}$$

where the matrix exponential is defined by

$$\exp(\tau \boldsymbol{A}) = \sum_{k=0}^{\infty} \frac{(\tau \boldsymbol{A})^k}{k!}. \tag{2.17}$$

The idea behind uniformization is to use a truncation of the series that avoids roundoff errors. To do so, the method applies the transformation

$$\boldsymbol{P} = \boldsymbol{I} + \frac{1}{\alpha}\boldsymbol{A}, \quad \alpha = \max_i |\boldsymbol{A}_{ii}|,$$

where $\boldsymbol{P}$ now has nonnegative entries, and then approximates (2.16) using

$$\begin{aligned} \exp(\tau \boldsymbol{A})\boldsymbol{v} &= \exp(\alpha\tau(\boldsymbol{P} - \boldsymbol{I}))\boldsymbol{v} \\ &= e^{-\alpha\tau} \exp(\alpha\tau \boldsymbol{P})\boldsymbol{v} \\ &\approx \sum_{k=0}^{\ell} e^{-\alpha\tau} \frac{(\alpha\tau)^k}{k!} \boldsymbol{P}^k \boldsymbol{v}. \end{aligned}$$

The last aspect of the uniformization method to consider is choosing the number of iterations ℓ. Noting that the way $\boldsymbol{P}$ is defined implies that $0 \leq \boldsymbol{P} \leq 1$ component-wise, and $\|\boldsymbol{P}\|_1 = 1$, from which it can be shown that

$$\left\| \exp(\tau \boldsymbol{A})\boldsymbol{v} - \sum_{k=0}^{\ell} e^{-\alpha\tau} \frac{(\alpha\tau)^k}{k!} \boldsymbol{P}^k \boldsymbol{v} \right\|_1 \leq \epsilon$$

if

$$1 - \sum_{k=0}^{\ell} e^{-\alpha\tau} \frac{(\alpha\tau)^k}{k!} \leq \epsilon.$$

In practice, the integration interval $[t_k, t_{k+1}]$ is usually subdivided to avoid overflow issues. This is often done by choosing a parameter θ ([25] suggests $\theta = 100$) and writing the solution as

$$m = \left\lceil \frac{\alpha\tau}{\theta} \right\rceil, \quad \bar{\tau} = \frac{\tau}{m},$$

$$\exp(\tau \boldsymbol{A})\boldsymbol{v} = \left(e^{-\alpha\bar{\tau}} \exp(\alpha\bar{\tau}\boldsymbol{P})\right)^m \boldsymbol{v},$$

then evaluating the last equation by starting from $\boldsymbol{\omega} = \boldsymbol{v}$ and iterating

$$\boldsymbol{\omega} \leftarrow e^{-\alpha\bar{\tau}} \exp(\alpha\bar{\tau}\boldsymbol{P})\boldsymbol{\omega}$$

m times.

Approximate by 2.7.1 Uniformization
(W. Grassmann, [26], 1977
D. Gross, D. Miller, [27], 1984)

0: Start with $\boldsymbol{p}_{k-1}(t_k)$, $\boldsymbol{A}_{J_k}$, τ_k and parameter θ
1: Initialize $\alpha = \max|\text{diag}(\boldsymbol{A}_{J_k})|$ and

$$\boldsymbol{P} = \boldsymbol{I} + \frac{\boldsymbol{A}_{J_k}}{\alpha}.$$

2: Find parameters $m = \left\lceil \frac{\alpha\tau}{\theta} \right\rceil$ and $\bar{\tau} = \frac{\tau_k}{m}$.
3: Choose ℓ so that

$$1 - \sum_{k=0}^{\ell} e^{-\alpha\bar{\tau}} \frac{(\alpha\bar{\tau})^k}{k!} \leq \epsilon.$$

4: Start with $\boldsymbol{\omega} = \boldsymbol{P}_{k-1}(t_k)$.
5: Iterate

$$\boldsymbol{\omega} \leftarrow \sum_{l=0}^{\ell} e^{-\alpha\bar{\tau}} \frac{(\alpha\bar{\tau})^l}{l!} \boldsymbol{P}^k \boldsymbol{\omega}$$

m times.
6: Return $\boldsymbol{p}_k(t_{k+1}) = \boldsymbol{\omega}$.

2.7.2 Approximate by Krylov-based techniques

One of the most effective methods to approximate the solution in the form of (2.16) is by using Krylov-based techniques. Given a vector $\boldsymbol{v}$ and matrix $\boldsymbol{A}$, we define the Krylov

subspace of order m to be

$$\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{v}) = span\{\boldsymbol{v}, \boldsymbol{A}\boldsymbol{v}, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{v}\}.$$

The well-known Arnoldi process produces an orthonormal basis $\boldsymbol{V}_m$ of $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{v})$, and an upper Hessenberg matrix $\boldsymbol{H}_m$. The Krylov approximation is then

$$\exp(\tau \boldsymbol{A})\boldsymbol{v} \approx \beta \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\boldsymbol{e}_1, \tag{2.18}$$

where $\beta = \|\boldsymbol{v}\|_2$ and $\boldsymbol{e}_1 = (1, 0, \ldots, 0)^T$.

We still need to approximate $\exp(\tau \boldsymbol{H}_m)$ in this equation. This can be done using the Padé approximation, together with scaling and squaring. This is a much cheaper problem than approximating $\exp(\tau \boldsymbol{A})$, since it has been shown in experiments that (2.18) yields a good approximation even when $m = 40$ or less.

The authors in [11] proposed integrating the Krylov method in the time-stepping FSP algorithm more than just to approximate $\boldsymbol{P}_k(t_{k+1})$. They did so by examining the local error

$$\Gamma = \mathbf{1}^T \boldsymbol{p}_k(t_{k+1})$$

and decreasing the step-size by half and reapplying Krylov method if the condition

$$\Gamma > 1 - \epsilon \frac{t_{k+1}}{t_f} \tag{2.19}$$

fails to be satisfied. The algorithm keeps halving the step-size until we have (2.19). Only then, the algorithm moves on to the new time point.

At the new time point, the algorithm only expands the state space if the time-step was reduced in the last time interval. This guarantees that the algorithm does not waste time finding and operating with a larger matrix unless it has to.

Approximate by 2.7.2 Krylov subspace

0: Start with the Krylov order m, $\boldsymbol{p}_{k-1}(t_k)$, $\boldsymbol{A}_{J_k}$, and an initial choice for τ_k.
1: Apply the Arnoldi process to find $\boldsymbol{V}_m$ and $\boldsymbol{H}_m$ for $\mathcal{K}_m(\boldsymbol{A}_{J_k}, \boldsymbol{p}_{k-1}(t_k))$.
2: Approximate the solution at $t_{k+1} = t_k + \tau_k$:

$$\boldsymbol{p}_k(t_{k+1}) \approx \beta \boldsymbol{V}_m \exp(\tau_k \boldsymbol{H}_m)\boldsymbol{e}_1,$$

where $\beta = \|\boldsymbol{p}_{k-1}(t_k)\|$.
$\exp(\tau_k \boldsymbol{H}_m)$ is approximated by the Padé approximation, with scaling and squaring.
3: $\Gamma = \mathbf{1}^T \boldsymbol{p}_k(t_{k+1})$.
4: Stop if $\Gamma > 1 - \epsilon \frac{t_{k+1}}{t_f}$ (see note), otherwise reduce τ_k by half and return to Step 2.

Note: For the next step in the time-stepping FSP algorithm, only expand the state space if halving happened in this step.

2.8 Tensor decomposition alternative

As will be shown in the example in Section 2.9, the biggest disadvantage of solving the CME is the "curse of dimensionality": the size of matrix $\boldsymbol{A}$ is great even when there are only a few species at play. In such cases, even if $\boldsymbol{A}$ is sparse, operating with it can be costly. In recent years, a lot of work has been done in decomposing the CME matrix using tensors, which promises a reduction in storage space.

In this section, we assume the FSP to take the form of a hyper-rectangle

$$\boldsymbol{X}_{J_k} = \{0, 1, \ldots, n_1\} \times \ldots \times \{0, 1, \ldots, n_N\}. \tag{2.20}$$

As a slight abuse of notation, we will use $\boldsymbol{A}$ instead of $\boldsymbol{A}_J$ to refer to the submatrix corresponding to the hyper-rectangle.

As a motivation, we note that under mild conditions the infinitesimal generator $\boldsymbol{A}$ can be decomposed into a sum of tensor products [28, 29, 30]

$$\boldsymbol{A} = \sum_{k=1}^{M} \otimes_{i=1}^{N} \boldsymbol{S}_k^i \boldsymbol{M}_k^i - \otimes_{i=1}^{N} \boldsymbol{M}_k^i, \tag{2.21}$$

where each term in the sum corresponds to a chemical reaction, the matrix $\boldsymbol{S}_k^i \in \mathbb{R}^{n_i \times n_i}$ is

the 'shifted-diagonal' matrix corresponding to the change in species i when reaction k happens, and the diagonal matrix $\boldsymbol{M}_k^i \in \mathbb{R}^{n_i \times n_i}$ that stores the values of i-factor in the propensity function α_k. This allows the matrix $\boldsymbol{A}$ to be stored in $O(MN \max(n_i))$ terms instead of $O(Mn_1 n_2 \dots n_N)$ terms of a straightforward sparse storage. Moreover, each constituent matrix can be subject to further compression techniques that improve further the memory management. We now turn to some tensor-based techniques that seek to compress the long probability vector $\boldsymbol{p}$.

The fact that $\boldsymbol{p}$ is indexed by the multi-dimensional states allows it to be reshaped into an $n_1 \times \ldots \times n_N$-dimensional array (or tensor), making the CME a natural target for tensor decompositions. The earliest attempt to apply tensor decomposition in the CME context that we know of is by Hegland and Garcke in 2011 [28], who sought approximations of the form

$$\boldsymbol{p} \approx \sum_{j=1}^{r} \otimes_{i=1}^{N} \boldsymbol{p}^i, \tag{2.22}$$

where the number of terms r is called the *rank* of the tensor decomposition and is made as small as possible for a prescribed tolerance. The approximation can be stored in $O(rN \max(n_i))$ and this avoids the curse of dimensionality if r is small. About one year later, Dolgov and Khoromskij proposed a different approach that used the Tensor Train (TT) format [31]. The TT decomposition breaks $\boldsymbol{p}$ into the 3-dimensional boxes $\boldsymbol{G}_0, \boldsymbol{G}_1, \dots, \boldsymbol{G}_N$ where $\boldsymbol{G}_i \in \mathbb{R}^{n_i \times r_i \times n_{i+1}}$ (with $n_0 = n_{N+1} = 1$). The numbers r_i are called the TT-ranks of $\boldsymbol{p}$ and each $\boldsymbol{G}_i$ a TT-core. The probability at any state $(x_1, \dots, x_N)$ is recovered by

$$\boldsymbol{p}(x_1, \dots, x_N) = \sum_{i_0=1}^{r_0} \dots \sum_{i_{N-1}=1}^{r_{N-1}} \boldsymbol{G}_0(i_0, x_1) \boldsymbol{G}_1(x_1, i_1, x_2) \dots \boldsymbol{G}_N(x_N, i_N),$$

which is an instance of *tensor contraction*. The TT approach reduces the $O(n_1 \dots n_N)$ storage of the full tensor $\boldsymbol{p}$ into $O(Nn^2 r_{tt})$ where $n = \max\{n_i\}$ and $r_{tt} = \max\{r_i\}$. If r_{tt} is small the compression rate is tremendous. The quantized tensor train (QTT) format used

in Kazeev et al. [30] takes the compression of the TT approach further by reshaping the already high-dimensional probability tensor $\boldsymbol{p}$ into an even higher-dimensional tensor with many virtual dimensions (as opposed to the physical dimensions represented by the chemical species). The TT decomposition is then applied on top of this reshaped tensor to achieve a higher compression rate. This is perhaps one of the most fascinating features of the tensor approaches that can potentially turn the curse of dimensionality into a blessing (to paraphrase [32]). Finally, we note that the techniques described in this paragraph can be applied equally to compress the matrix $\boldsymbol{A}$ itself to potentially overcome the memory explosion issue in solving the CME.

So far we have only mentioned the compression strategies for the large matrix and vector in the CME using tensor decompositions. The challenge is to design numerical schemes that maintain the benefits brought by these techniques. Unfortunately, classical matrix methods do not lend themselves easily to the new formats. The work of Dolgov in adapting the GMRES method to TT format reveals incompatibility between Krylov subspace methods and TT decomposition: the TT-ranks of the Krylov vectors given by the Arnoldi iteration increase even though both $\boldsymbol{A}$ and $\boldsymbol{p}$ have low TT-ranks. There are, however, promising tools being developed and analyzed for the tensor format such as the Density matrix renormalization group (DMRG) solver for linear systems in tensor format. Based on this, implicit time-stepping schemes can be employed to integrate the CME. This is essentially the approach of Kazeev et al. [30], where the *hp-discontinuous Galerkin* scheme is applied successfully in many non-trivial CME problems in quantized TT-format. Alternatively, classical time-stepping schemes like implicit Euler can be used to form a global linear system in tensor format to solve once and for all for the snapshots of the time-dependent probability distribution as done by [33]. Such scheme would have been costly in traditional matrix-vector format, but becomes much more feasible in tensor format due to its strong compression ability. We refer to the numerical results in the cited paper that show the prospects of this new approach.

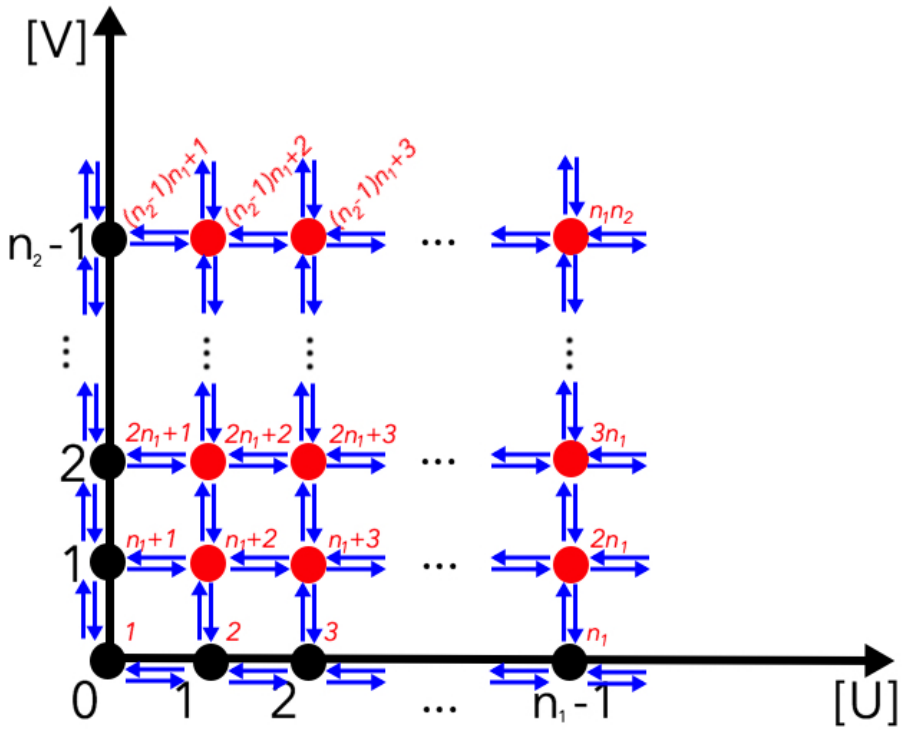


Figure 2.1: A lattice describing all possible states for the stochastic gene toggle model, in Example 1. The limits in the FSP for protein U is n_1, and protein V is n_2. The red numbers are the state indices.

The tensor decomposition approaches to the CME are just a wave in the growing currents of tensor techniques with wide applications in different fields of science. We refer to the reviews by [34, 33] and [32].

2.9 Illustrative example

We will consider the stochastic gene toggle model, which is a simple model usually employed for its interesting properties. This example is meant to showcase how the matrix used in the CME is generated. There are two species in the model, U and V, and the production of each has a negative feedback on the production of the other species. The

four reactions that U and V participate in and their propensities are: [1, 35]

$$\begin{aligned}
\varnothing \xrightarrow{k_1} \mathrm{U}: &\quad \alpha_1 = k_1 \\
\mathrm{U} \xrightarrow{k_2} \varnothing: &\quad \alpha_2 = k_2[\mathrm{U}] \\
\varnothing \xrightarrow{k_3} \mathrm{V}: &\quad \alpha_3 = k_3 \\
\mathrm{V} \xrightarrow{k_4} \varnothing: &\quad \alpha_4 = k_4[\mathrm{V}]
\end{aligned}$$

The parameters, taken from [36], showcase the feedback between the two species: $k_1 = 0.2 + \frac{4}{1+[\mathrm{V}]^3}, k_2 = 1.09, k_3 = 0.2 + \frac{4}{1+[\mathrm{U}]^3}, k_4 = 1.$ k_1 and k_3 can be thought of as functions of [V] and [U], respectively. We set the initial state as $\boldsymbol{x}_0 = ([\mathrm{U}], [\mathrm{V}]) = (85, 5)$.

Since the numbers of U and V can be any nonnegative numbers, we need a bound so that the FSP matrix is finite dimensional. Let $n_1 - 1$ and $n_2 - 1$ be the upper bounds on [U] and [V], so there are a total of $n_1 n_2$ states that we keep track of (because [U] and [V] can be 0).

Each state of the system consists of the numbers of proteins U and V. To identify the state with a single number, we need the following indexing formula:

$$i([\mathrm{U}], [\mathrm{V}]) = [\mathrm{U}] + 1 + n_1[\mathrm{V}].$$

The states and reactions are shown in Figure 2.1. The state index $i([\mathrm{U}], [\mathrm{V}])$ is shown in red.

We will now formulate the ODE describing the evolution of the probabilities over time. The vector form of the ODE is

$$\dot{\boldsymbol{p}}(t) = \boldsymbol{A} \cdot \boldsymbol{p}(t),$$

where matrix $\boldsymbol{A} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ has the block form of

$$\begin{bmatrix} \boldsymbol{D}_0 & \boldsymbol{C}_1 & \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{D}_1 & \boldsymbol{C}_2 & \dots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B} & \boldsymbol{D}_2 & \dots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & & \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & & \boldsymbol{D}_{n_2-2} & \boldsymbol{C}_{n_2-1} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & & \boldsymbol{B} & \boldsymbol{D}_{n_2-1} \end{bmatrix}$$

where

$$\boldsymbol{B} = \text{diag}(k_3(0), \dots, k_3(n_1 - 1)) \in \mathbb{R}^{n_1 \times n_1}$$

accounts for the production of V, which can be thought of as an upward transition in Figure 2.1. On the other hand,

$$\boldsymbol{C}_m = \text{diag}(m \cdot k_4, \dots, m \cdot k_4) \in \mathbb{R}^{n_1 \times n_1}$$

describes the downward transition of states in row m of Figure 2.1, through the death of one protein V.

Finally, the diagonal blocks are computed as

$$\begin{aligned} \boldsymbol{D}_0 &= \boldsymbol{T}_0 - \boldsymbol{B} \\ \boldsymbol{D}_m &= \boldsymbol{T}_m - \boldsymbol{B} - \boldsymbol{C}_m, \quad m \geq 1, \end{aligned}$$

where $\boldsymbol{T}_m$ depicts the left and right transitions within row m of Figure 2.1, through the

birth or death of one protein U:

$$\boldsymbol{T}_m(i,j) = \begin{cases} -k_1(m) & i = j = 1 \\ -k_1(m) - (j-1)\cdot k_2 & i = j > 1 \\ (j-1)\cdot k_2 & j = i+1 \\ k_1(m) & j = i-1 \end{cases}$$

The gene toggle model is well-known for its bistability: there are two different stable steady modes that the system can converge to, which can be seen easily using the SSA-driven FSP method [1], shown in the last row of Figure 2.2.

The first row in Figure 2.2 shows the distributions from applying the SSA with 10^5 trajectories. All of the trajectories converge to only one of the two steady states. The reason is transparent: the result from the SSA-driven FSP method informs us that the probabilities of the states in the second mode are between 10^{-8} and 10^{-13}. Therefore many more trajectories would be needed for the SSA to reach bimodality. This is shown more clearly when 10^8 trajectories are simulated by the SSA, as shown in the second row of Figure 2.2.

Therefore this example shows that in some cases, solving the CME by using the FSP can be both more accurate and much faster than applying the SSA or other stochastic methods.

2.10 Softwares

Here we discuss some softwares and packages that illustrate the methodology of the FSP and serve as an efficient means to solve the CME.

2.10.1 FSP Toolkit

FSP Toolkit is a MATLAB software for solving the CME for two species using the FSP, which can be found at

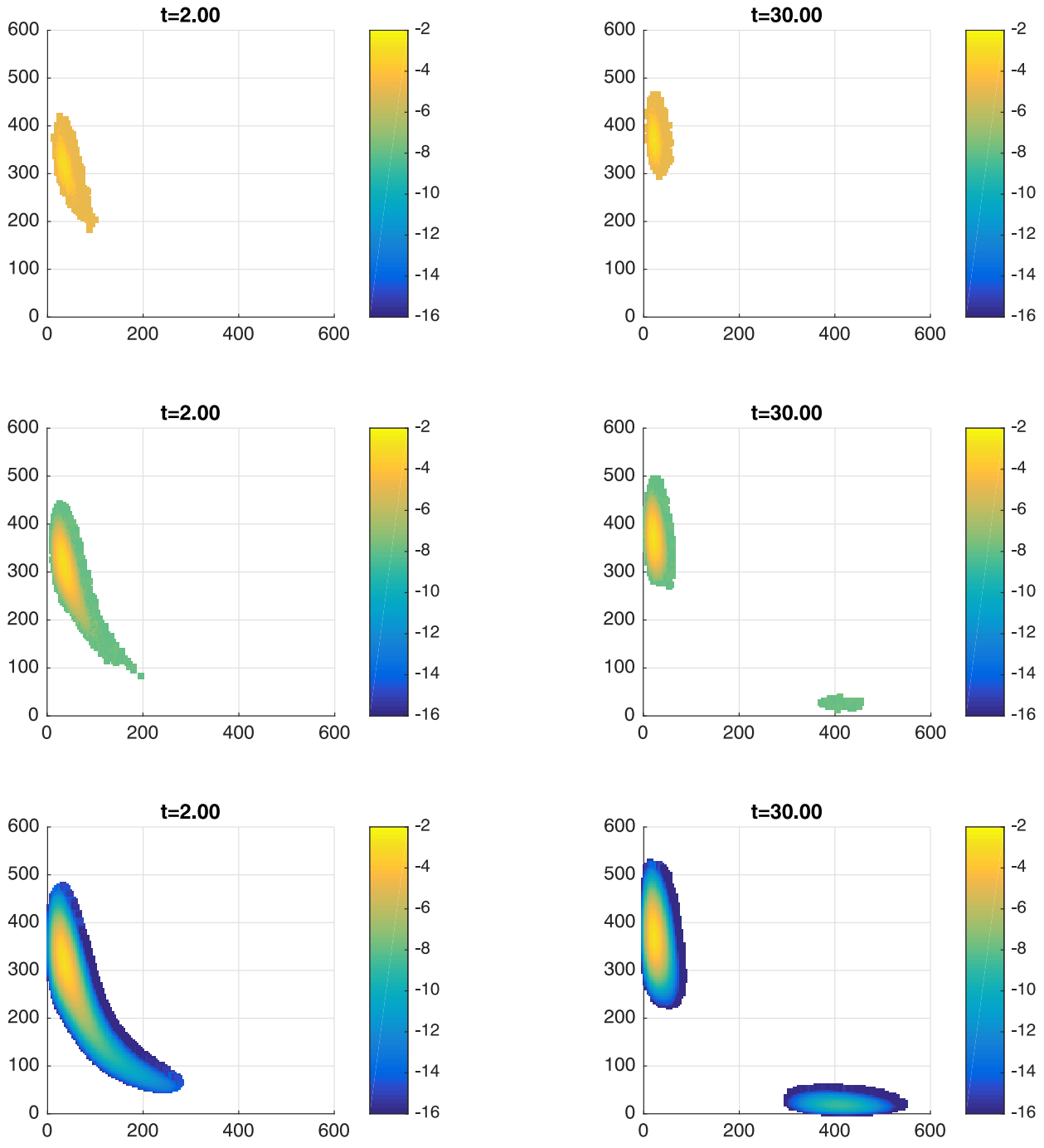


Figure 2.2: Probability distributions (logscale) of the stochastic gene toggle after 2s (left column) and 30s (right column) using SSA with 10^5 trajectories (first row), SSA with 10^8 trajectories (second row), and SSA-driven FSP [1] (third row).

http://cnls.lanl.gov/∼munsky/Software.html

Ref. [18] describes the numerical method in detail. The initial state space is defined by a set of nonlinear inequalities, and it is expanded by applying r-step reachability and multiple absorbing states. A number of stochastic phenomena involving two species in biological systems are illustrated in the software, including activation through linear regulation, activation with a convex or concave function, and toggle switch. The toolkit is very well explained and therefore recommended as a valuable resource for people new to the FSP approach.

2.10.2 CMEPy

CMEPy is a Python package for solving the CME, which can be downloaded and installed at

http://fcostin.github.io/cmepy/index.html

The program expands the state space by r-step reachability. It can also solve for the case where the propensities are time-dependent, but only when a separation of variables can be applied:

$$\alpha_k(\boldsymbol{x}, t) = \phi_k(\boldsymbol{x})\theta(t).$$

2.10.3 Expokit

As mentioned in Section 2.7, Expokit is one of the most efficient software to calculate the matrix exponential of either small dense or very large sparse matrices. The package is written in Fortran and MATLAB [37], and can be found at

http://www.maths.uq.edu.au/expokit/

It is the basis of ongoing solution techniques of the FSP.

2.11 Discussion

The FSP is an especially effective method in a number of gene expression regulation problems for several reasons. First of all, there are few species involved and upper limits on these species numbers are usually given either from experimental data or theoretical biology, implying that the size of CME may occasionally be manageable. Secondly, the goal in these problems is usually to find the model that explains the experimental data, and to find the model parameter (i.e. a vector of reaction rates) that results in the probability distributions that best fit the data at different times. Since thousands or even millions of different model parameters have to be computed and compared, the probability distributions have to be solved efficiently and fast, in which case the FSP has an advantage over kinetic Monte Carlo simulations, which require large numbers of simulations. We refer to [38] which contains a number of good examples where the FSP is applied in real-life gene expression regulation problems.

However, the "curse of dimensionality" makes solving the CME numerically difficult if not impossible in the case where there are many species. An example for such a case is the regulation of protein p53 [39], where there are six species of interest, interacting with each other through 11 reactions. A bound B is set to be the maximum number of molecules of each species that the cell can contain. It is then obvious that the number of states that the system can be in is roughly B^6, where B can be thousands. We then end up with a very large, although sparse, matrix $\boldsymbol{A}$ for the CME. In practice, when we solve the CME using real parameters as reported in [39], the probability mass requires a projection of over 4 million states and at each time point, expanding the state space to the next time step would explode the projection up to over 15 million states.

When such a huge system is encountered, the FSP method fails and the best numerical methods to employ are the SSA and other Monte Carlo methods.

2.12 Conclusion

The amount of research efforts in the last few years that built on the finite state projection method is the most convincing evidence of the importance of the method in solving the chemical master equation. Variants of the original algorithm have led to tremendous improvements. This chapter offered a review of these methods in a systematic fashion. We outlined the core ideas behind the variants, and highlighted similarities and differences between them. We note that in addition to biological applications, the FSP is found useful in other areas as well [40].

CHAPTER 3

ANALYSIS OF INEXACT KRYLOV SUBSPACE METHODS

This chapter is based on work published in Mathematics and Computers in Simulation [41].

3.1 Introduction

Here, we seek to generalize the Finite State Projection and develop theoretical background for this reduction technique.

Given a large sparse nonsymmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and vector $\boldsymbol{p}_0 \in \mathbb{R}^n$, letting $\boldsymbol{v} = \boldsymbol{p}_0$ and taking $m \ll n$ Arnoldi steps with a starting vector $\boldsymbol{v}_1 = \boldsymbol{v}/\|\boldsymbol{v}\|$, where $\|\cdot\|$ means the 2-norm, we obtain an orthonormal basis $\boldsymbol{V}_m = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m] \in \mathbb{R}^{n\times m}$ of the Krylov subspace $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{v}) = \text{span}\{\boldsymbol{v}, \boldsymbol{A}\boldsymbol{v}, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{v}\}$, and an upper Hessenberg matrix $\boldsymbol{H}_m \in \mathbb{R}^{m\times m}$ that satisfy

$$\boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{V}_{m+1}\overline{\boldsymbol{H}}_m = \boldsymbol{V}_m\boldsymbol{H}_m + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T, \tag{3.1a}$$

$$\boldsymbol{H}_m = \boldsymbol{V}_m^T\boldsymbol{A}\boldsymbol{V}_m, \tag{3.1b}$$

where $\boldsymbol{e}_m = (0, \ldots, 0, 1)^T$, and $\overline{\boldsymbol{H}}_m \in \mathbb{R}^{(m+1)\times m}$ is $\boldsymbol{H}_m$ augmented with $h_{m+1,m}\boldsymbol{e}_m^T$ under its last row. The standard Krylov approximation to the matrix exponential takes the form

$$\exp(\tau\boldsymbol{A})\boldsymbol{v} \approx \boldsymbol{V}_m \exp(\tau\boldsymbol{H}_m)\beta\boldsymbol{e}_1, \quad \boldsymbol{e}_1 = (1, 0, \ldots, 0)^T, \quad \beta = \|\boldsymbol{v}\|. \tag{3.2}$$

It is well-known that (3.1) is also the cornerstone for building very efficient Krylov subspace solution techniques for other problems such as eigenvalue problems or linear systems. In the latter, there has been recent interest in transitioning from exact to inexact

(or relaxed) matrix-vector products in the Arnoldi process [42, 43, 44], either out of necessity or deliberately, trading accuracy for speed. It is customary to model these inexact products as

$$\boldsymbol{A}\boldsymbol{v}_k \approx (\boldsymbol{A} + \boldsymbol{E}_k)\boldsymbol{v}_k, \tag{3.3}$$

where $\boldsymbol{E}_k$ is some error matrix that varies at each invocation, and note that setting $\boldsymbol{E}_k = \boldsymbol{0}$ recovers the exact evaluation. To make the difference clear, we refer to the classical method as the exact Arnoldi and it is not meant to imply exact arithmetic. The foremost implication of such a relaxation is that the classical Arnoldi relationship (3.1) does not hold anymore, but Simoncini and Szyld [44] made the key observation that we end up with

$$(\boldsymbol{A} + \boldsymbol{\mathcal{E}}_m)\boldsymbol{V}_m = \boldsymbol{V}_m\boldsymbol{H}_m + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T, \qquad \boldsymbol{\mathcal{E}}_m = \sum_{k=1}^{m} \boldsymbol{E}_k\boldsymbol{v}_k\boldsymbol{v}_k^T, \tag{3.4}$$

which is similar to (3.1), except that $\boldsymbol{V}_m$, which still remains orthonormal, is now a basis of a Krylov subspace obtained by a perturbed $\boldsymbol{A}$. When we use the computed $\boldsymbol{V}_m$ and $\boldsymbol{H}_m$ from (3.4) in GMRES for instance, classical error bounds do not apply anymore. However, from theoretical and experimental evidence (such as [45]), the method can withstand cases where the norm of the perturbation $\boldsymbol{\mathcal{E}}_m$ grows very large.

The analysis of Simoncini and Szyld [44] provided insights into inexact GMRES for solving a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, but it has so far remained unclear how inexactness affects the Krylov approximation (3.2). Since we now have (3.4) instead of (3.1), we also lose classical error bounds on the matrix exponential (e.g., Gallopoulos and Saad [46], Saad [47], Hochbruch and Lubich [48]). Thus our study here fills a gap in the literature by looking at the error in the inexact Krylov counterpart of (3.2). We additionally offer another related way of assessing the accuracy by investigating the defect or residual [49]

from the fact that (3.2) arises when solving a system of linear ODEs of the form

$$\begin{cases} \boldsymbol{p}'(t) = \boldsymbol{A}\boldsymbol{p}, \quad t \in [0,\, t_f] \\ \boldsymbol{p}(0) = \boldsymbol{p}_0, \quad \text{initial condition.} \end{cases} \tag{3.5}$$

It is worth recalling that, in the exact case, the effectiveness of approximating $\exp(\boldsymbol{A})\boldsymbol{v}$ by projecting it onto $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{v})$ hinges on the fact that all polynomials of $\boldsymbol{A}$ of degree $\leq m-1$ can be calculated exactly through $\boldsymbol{H}_m$, or more precisely,

$$q_{m-1}(\boldsymbol{A})\boldsymbol{v} = \boldsymbol{V}_m q_{m-1}(\boldsymbol{H}_m)\beta\boldsymbol{e}_1,$$

where q_{m-1} is any polynomial of degree $\leq m-1$. By the same reasoning as in the exact case (e.g., Saad [47, Lemma 3.1]), it can be shown from (3.4) that, for the same polynomial q_{m-1},

$$q_{m-1}(\boldsymbol{A} + \boldsymbol{\mathcal{E}}_m)\boldsymbol{v} = \boldsymbol{V}_m q_{m-1}(\boldsymbol{H}_m)\beta\boldsymbol{e}_1.$$

The implication of all this is that the inexact Krylov subspace method for $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{v})$ with the relaxation matrices $\boldsymbol{E}_k, k = 1, \ldots, m$, can be seen as the exact Krylov subspace method for $\mathcal{K}_m(\tilde{\boldsymbol{A}}, \boldsymbol{v})$ with

$$\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{\mathcal{E}}_m = \boldsymbol{A} + \sum_{k=1}^{m} \boldsymbol{E}_k \boldsymbol{v}_k \boldsymbol{v}_k^T,$$

which is another simple way to understand the method.

The rest of the chapter is organized as follows: Section 3.2 gives some background on modeling biochemical reactions. Section 3.3 analyzes the residual (or defect) of the inexact Krylov method both when the ODE problem is homogeneous or nonhomogeneous, with two different approaches considered for the latter. Section 3.4 analyzes the error and includes a special treatment that exploits the structure of the matrix when it arises from stochastic processes such as that involved in the CME. Section 3.5 reports some numerical

experiments. Section 3.6 finally wraps the presentation with some concluding remarks.

3.2 Inexact chemical master equation - a motivation

In a biological cell containing different molecular species undergoing various chemical reactions, the state is a vector of integers counting the different species of molecules. Such a discrete formulation is prompted by key regulatory molecules that exist in small numbers, making a continuous formulation (via concentrations) inappropriate. The counter of a species goes up or down when a chemical reaction occurs, depending on whether the reaction produces or consumes that species. Starting from a particular state, the cell will transition to different states as reactions happen. The CME depicts the evolution of the system's probability distribution, which characterizes the probability of finding the cell in a given state at a given time. The challenge here is that even with simple biochemical models having 4 or 5 reaction channels and a relatively low count of each molecular species, there can be millions of possible states, and the variety of models means that calculations cannot rely on generic simplifications.

The finite state projection (FSP) algorithm [10] can be thought of as a model reduction method to cope with the huge size of the CME, and we outline its matrix form here because it vividly illustrates how inexactness comes into play. With $J = \{1, \ldots, k\}$, and k being the cardinality of J, let

$$\boldsymbol{A} = \left(\begin{array}{c|c} \boldsymbol{A}_J & * \\ \hline * & * \end{array} \right) \in \mathbb{R}^{n \times n},$$

i.e., $\boldsymbol{A}_J$ is a $k \times k$ submatrix of the true operator $\boldsymbol{A}$. The FSP algorithm takes

$$\boldsymbol{p}(t_f) = \exp(t_f \boldsymbol{A}) \boldsymbol{p}_0 \approx \begin{pmatrix} \exp(t_f \boldsymbol{A}_J) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_J(0) \\ \boldsymbol{0} \end{pmatrix}. \tag{3.6}$$

Note that $\boldsymbol{p}_0 = \boldsymbol{p}(0)$ is also truncated according to J. Munsky and Khammash [10]

assessed the loss of the probability mass and gave a theoretical justification of the merit of this approach from a probabilistic point of view. We now recast the analysis in the framework of inexact methods.

With J now an arbitrary subset of $\{1, \ldots, n\}$, and the corresponding submatrix $\boldsymbol{A}_J$ padded with zeros as necessary, the 'truncation' can be formalized as

$$\boldsymbol{A} = \boldsymbol{A}_J + \boldsymbol{R}_J,$$

where $\boldsymbol{R}_J$ is the error matrix from the truncation. Then (3.6) is in turn further approximated by an inexact Krylov method for the matrix exponential, which we saw previously means that the Arnoldi process for approximating the solution uses inexact (or relaxed) matrix-vector products of the form

$$\boldsymbol{A}_J\boldsymbol{v} \approx (\boldsymbol{A}_J + \boldsymbol{E}_J)\boldsymbol{v},$$

where $\boldsymbol{E}_J$ models some error in the product. Combining with the truncation, we get

$$\boldsymbol{A}\boldsymbol{v} \approx (\boldsymbol{A} - \boldsymbol{R}_J + \boldsymbol{E}_J)\boldsymbol{v},$$

so that $\boldsymbol{E} = -\boldsymbol{R}_J + \boldsymbol{E}_J$ captures both error terms. In particular, if $\boldsymbol{E}_J = 0$, only the truncation error is in effect, whereas if $\boldsymbol{R}_J = 0$, there will only be the error from the relaxed matrix-vector product. From now on, our theoretical analysis will simply assume that the true matrix $\boldsymbol{A}$ interacts through the inexact evaluation $\boldsymbol{A}\boldsymbol{v}_k \approx (\boldsymbol{A} + \boldsymbol{E}_k)\boldsymbol{v}_k$ without regard to the real source of the error.

3.3 Bounds on the residual

Given the differential equation (3.5), consider an approximation $\boldsymbol{p}_m(t) \approx \boldsymbol{p}(t)$, then in the terminology of ODEs the 'residual' or 'defect' is defined as

$$\boldsymbol{r}_m(t) = \boldsymbol{p}'_m(t) - \boldsymbol{A}\boldsymbol{p}_m(t). \tag{3.7}$$

This definition reminds what happens when approximating the solution to a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Take GMRES, which uses the approximation $\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V}_m\boldsymbol{y}_m$, where $\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{y}_m = \overline{\boldsymbol{H}}_m^{\dagger}\beta\boldsymbol{e}_1$, with $\boldsymbol{V}_m$ and $\overline{\boldsymbol{H}}_m$ arising from the Arnoldi process for $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{r}_0)$, and denoting $r_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\beta = \|\boldsymbol{r}_0\|$, and $\overline{\boldsymbol{H}}_m^{\dagger}$ the pseudo-inverse of $\overline{\boldsymbol{H}}_m$. Applying (3.4), the inexact GMRES method ends up with (the norm of) the *computed* residual

$$\tilde{\boldsymbol{r}}_m = \boldsymbol{r}_0 - \boldsymbol{V}_{m+1}\overline{\boldsymbol{H}}_m\boldsymbol{y}_m, \tag{3.8}$$

while the *true* residual satisfies

$$\boldsymbol{r}_m = \boldsymbol{b} - \boldsymbol{A}(\boldsymbol{x}_0 + \boldsymbol{V}_m\boldsymbol{y}_m) = \tilde{\boldsymbol{r}}_m + \boldsymbol{\mathcal{E}}_m\boldsymbol{V}_m\boldsymbol{y}_m. \tag{3.9}$$

From (3.8) and (3.9), and the fact that $\boldsymbol{\mathcal{E}}_m\boldsymbol{V}_m = [\boldsymbol{E}_1\boldsymbol{v}_1, \ldots, \boldsymbol{E}_m\boldsymbol{v}_m]$ because $\boldsymbol{V}_m$ is orthonormal, there is an unknown *residual gap* that satisfies

$$\delta_m = \|\boldsymbol{r}_m - \tilde{\boldsymbol{r}}_m\| = \|\boldsymbol{\mathcal{E}}_m\boldsymbol{V}_m\boldsymbol{y}_m\| = \|[\boldsymbol{E}_1\boldsymbol{v}_1, \ldots, \boldsymbol{E}_m\boldsymbol{v}_m]\boldsymbol{y}_m\|. \tag{3.10}$$

The inexact solver still reports $\|\tilde{\boldsymbol{r}}_m\|$ as the estimate of the residual, but the residual gap (3.10) is not obvious, and so the reliability of the final solution is not guaranteed. Simoncini and Szyld [44] obtained the bound (see also [50, 45]):

$$\delta_m \le \sum_{k=1}^{m} |\eta_k^{(m)}| \cdot \|\boldsymbol{E}_k\boldsymbol{v}_k\| \le \sum_{k=1}^{m} |\eta_k^{(m)}| \cdot \|\boldsymbol{E}_k\|, \quad \boldsymbol{y}_m = \overline{\boldsymbol{H}}_m^{\dagger}\beta\boldsymbol{e}_1 = (\eta_1^{(m)}, \eta_2^{(m)}, \ldots, \eta_m^{(m)})^T.$$

Returning to the differential equation (3.5) where the residual of an approximation is defined by (3.7), and using (3.2) based on the exact Arnoldi process, we get the Krylov approximation $\boldsymbol{p}_m(\tau) = \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1$ that leads to

$$\begin{aligned}
\boldsymbol{p}'_m(\tau) - \boldsymbol{A}\boldsymbol{p}_m(\tau) &= \boldsymbol{p}'_m(\tau) - \boldsymbol{A}\boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 \\
&= \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\boldsymbol{H}_m \beta \boldsymbol{e}_1 - [\boldsymbol{V}_m \boldsymbol{H}_m + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T] \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 \\
&= -\beta h_{m+1,m}\Big(\boldsymbol{e}_m^T \exp(\tau \boldsymbol{H}_m) e_1\Big)\boldsymbol{v}_{m+1}.
\end{aligned}$$

Note that it is a variant of this estimate (with some scaling) that Expokit uses so far to great effect to monitor the accuracy. We define the *computed* residual as

$$\tilde{\boldsymbol{r}}_m(\tau) = -\beta h_{m+1,m}\Big(\boldsymbol{e}_m^T \exp(\tau \boldsymbol{H}_m) e_1\Big)\boldsymbol{v}_{m+1},$$

because this would still be the economical quantity (or a related one from it) used to estimate the *true* residual. As we shall see in the following section, with the inexact Arnoldi process (3.4), the *true* residual in the ODE problem becomes

$$\boldsymbol{r}_m(\tau) = \tilde{\boldsymbol{r}}_m(\tau) + \boldsymbol{\mathcal{E}}_m \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1,$$

which is reminiscent of (3.9), but with $\boldsymbol{y}_m(\tau) = \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1$ instead of the least squares solution $\boldsymbol{y}_m = \overline{\boldsymbol{H}}_m^{\dagger}\beta \boldsymbol{e}_1$. Simoncini and Szyld [44] refined their bounds by exploiting properties satisfied by the components of $\boldsymbol{y}_m$ through Givens rotations in the case of GMRES. Our analysis will derive bounds without assuming those properties since Givens rotations are not involved in the case of the matrix exponential.

3.3.1 Homogeneous case

The Krylov technique for solving (3.5) is typically done by using the integration scheme

$$\begin{cases} \boldsymbol{p}(0) = \boldsymbol{p}_0 \\ \boldsymbol{p}(t_{k+1}) = \exp(\tau_k \boldsymbol{A})\boldsymbol{p}(t_k), \end{cases} \tag{3.11}$$

with some strategy for chosing the stepsizes $\tau_k = t_{k+1} - t_k$. The problem remains how to effectively approximate $\exp(\tau \boldsymbol{A})\boldsymbol{v}$ given τ and $\boldsymbol{v}$.

Now, using (3.2) based on the inexact Arnoldi process (3.4), the *true* residual of the resulting Krylov approximation $\boldsymbol{p}_m(\tau) = \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1$ satisfies

$$\begin{aligned} \boldsymbol{r}_m &= \boldsymbol{p}_m' - \boldsymbol{A}\boldsymbol{p}_m \\ &= (\boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1)' - \boldsymbol{A}\boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 \\ &= \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\boldsymbol{H}_m \beta \boldsymbol{e}_1 - (\boldsymbol{V}_m \boldsymbol{H}_m + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T - \boldsymbol{\mathcal{E}}_m \boldsymbol{V}_m)\exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 \\ &= -h_{m+1,m}(\boldsymbol{e}_m^T \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1)\boldsymbol{v}_{m+1} + \boldsymbol{\mathcal{E}}_m \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 \\ &= \tilde{\boldsymbol{r}}_m + \boldsymbol{\mathcal{E}}_m \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1, \end{aligned}$$

where $\tilde{\boldsymbol{r}}_m$ is the computed residual defined earlier. The dependency on the time τ will not be made explicit unless there is a risk of ambiguities. The quantity

$$\delta_m^{\text{res}} = \|\boldsymbol{r}_m - \tilde{\boldsymbol{r}}_m\| = \|\boldsymbol{\mathcal{E}}_m \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1\| \tag{3.12}$$

is the *residual gap* between $\boldsymbol{r}_m$ and $\tilde{\boldsymbol{r}}_m$. It depends on the relaxation matrices $\boldsymbol{E}_k$ and therefore cannot be computed in a straightforward manner. From the derivation above, we

can bound the norm of the true residual as

$$\|\boldsymbol{r}_m\| \leq \|\tilde{\boldsymbol{r}}_m\| + \delta_m^{\text{res}}. \tag{3.13}$$

When all matrix-vector products are exact, i.e., all $\boldsymbol{E}_k = \boldsymbol{0}$, then $\boldsymbol{\mathcal{E}}_m = \boldsymbol{0}$ and $\delta_m^{res} = 0$ as expected. Therefore with $\boldsymbol{y}_m = \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1$, the residual gap can be written as

$$\delta_m^{\text{res}} = \|\boldsymbol{\mathcal{E}}_m \boldsymbol{V}_m \boldsymbol{y}_m\| = \|[\boldsymbol{E}_1 \boldsymbol{v}_1, \ldots, \boldsymbol{E}_m \boldsymbol{v}_m]\boldsymbol{y}_m\|,$$

and similarly to [44, Prop. 4.1], if we write $\boldsymbol{y}_m = (\eta_1^{(m)}, \ldots, \eta_m^{(m)})^T$, the following upper bound on the residual gap holds:

$$\delta_m^{\text{res}} \leq \sum_{k=1}^{m} |\eta_k^{(m)}| . \|\boldsymbol{E}_k\|, \tag{3.14}$$

and furthermore we have the following result:

Proposition 3.3.1. *Given $\epsilon > 0$, if $\|\boldsymbol{E}_k\| \leq \dfrac{\epsilon^{res}}{m\beta \|\exp(\tau \boldsymbol{H}_m)\|}, k = 1, \ldots, m$, then we have*

$$\delta_m^{res} \leq \epsilon^{res} \tag{3.15}$$

and therefore $\|\boldsymbol{r}_m\| \leq \|\tilde{\boldsymbol{r}}_m\| + \epsilon^{res}$.

Proof. We have $|\eta_k^{(m)}| \leq \|\boldsymbol{y}_m\| = \|\exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1\| \leq \beta \|\exp(\tau \boldsymbol{H}_m)\|$, and so combining the condition on $\|\boldsymbol{E}_k\|$ with the inequality (3.14) gives (3.15). The bound on $\|\boldsymbol{r}_m\|$ then follows naturally from the relation $\|\boldsymbol{r}_m\| \leq \|\tilde{\boldsymbol{r}}_m\| + \delta_m^{\text{res}}$. □

Remark 3.3.2. As Giraud et al. [50] pointed out in the context of inexact GMRES, Proposition 3.3.1 does not allow us to anticipate the relaxation matrices $\boldsymbol{E}_k$ in advance because of the dependence on $\boldsymbol{H}_m$. It can however be used in a postmortem manner to check if the error condition is satisfied.

3.3.2 Nonhomogeneous case

Bounding the residual gap via the φ function

It is well known (see, e.g., Expokit [37]) that the numerical solution to the system of nonhomogeneous ODEs

$$\begin{cases} \boldsymbol{p}'(t) = \boldsymbol{A}\boldsymbol{p}(t) + \boldsymbol{b} \\ \boldsymbol{p}(0) = \boldsymbol{p}_0 \end{cases} \tag{3.16}$$

with constant $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$, can be found using the integration scheme

$$\begin{cases} \boldsymbol{p}(0) = \boldsymbol{p}_0 \\ \boldsymbol{p}(t_{k+1}) = \tau_k \varphi(\tau_k \boldsymbol{A})[\boldsymbol{A}\boldsymbol{p}(t_k) + \boldsymbol{b}] + \boldsymbol{p}(t_k) \end{cases} \tag{3.17}$$

where $\varphi(\tau\boldsymbol{A}) = \sum_{i=0}^{\infty} \frac{(\tau\boldsymbol{A})^i}{(i+1)!}$ [51, Chap. 2.1, Chap. 10.7]. This integration scheme circumvents using the representation of the analytical solution of (3.16), $\boldsymbol{p}(t) = \exp(t\boldsymbol{A})\boldsymbol{p}_0 + t\varphi(t\boldsymbol{A})\boldsymbol{b}$, which would need both $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{p}_0)$ and $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{b})$ instead of only one Krylov subspace as implied in (3.17). Using this scheme, we can derive a result similar to Proposition 3.3.1, but this can be avoided by the augmented approach shown below.

Bounding the residual gap via an augmented matrix exponential

An indirect way to solve (3.16) takes root in the analytical solution $\boldsymbol{p}(t) = \exp(t\boldsymbol{A})\boldsymbol{p}_0 + t\varphi(t\boldsymbol{A})\boldsymbol{b}$, and has proved convenient in other circumstances such as [52, 53]. Define the augmented matrix

$$\boldsymbol{A}^+ = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{0} & 0 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)},$$

then we have

$$\exp(t\boldsymbol{A}^+) = \begin{pmatrix} \exp(t\boldsymbol{A}) & t\varphi(t\boldsymbol{A})\boldsymbol{b} \\ \boldsymbol{0} & 1 \end{pmatrix},$$

so that the solution can be fetched as $\boldsymbol{p}(t) = \left[\exp(t\boldsymbol{A}^+)\boldsymbol{p}_0^+\right]_{1:n}$ with $\boldsymbol{p}_0^+ = \begin{pmatrix} \boldsymbol{p}_0 \\ 1 \end{pmatrix}$.

The problem now amounts to getting $\exp(t\boldsymbol{A}^+)\boldsymbol{p}_0^+$. Transforming the problem back to the form $\exp(t\boldsymbol{A}^+)\boldsymbol{p}_0^+$ not only inherits the analysis done in the homogeneous case in an elegant way, but also enables seamless code re-use. Furthermore, the inexactness in the matrix-vector product with $\boldsymbol{A}^+$ is only triggered from $\boldsymbol{A}$ through

$$\boldsymbol{A}^+ + \boldsymbol{E}_k^+ = \begin{pmatrix} \boldsymbol{A} + \boldsymbol{E}_k & \boldsymbol{b} \\ \boldsymbol{0} & 0 \end{pmatrix},$$

and so results can nicely be recast in terms of $\boldsymbol{E}_k$. For this reason, we will not dwell any further on the nonhomogeneous case in the rest of our presentation.

3.4 Bounds on the error

3.4.1 General upper bound on the error

As pointed out in the introduction, the inexact method for $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{v})$ with the perturbation matrices $\boldsymbol{E}_k, k = 1, \ldots, m$, can be seen as the exact method for $\mathcal{K}_m(\tilde{\boldsymbol{A}}, \boldsymbol{v})$ with

$$\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{\mathcal{E}}_m = \boldsymbol{A} + \sum_{k=1}^{m} \boldsymbol{E}_k \boldsymbol{v}_k \boldsymbol{v}_k^T.$$

Therefore if we define

$$\begin{aligned} \tilde{\epsilon}_m &= \| \exp(\tau\tilde{\boldsymbol{A}})\boldsymbol{v} - \boldsymbol{V}_m \exp(\tau\boldsymbol{H}_m)\beta\boldsymbol{e}_1 \| \\ &= \| \exp(\tau(\boldsymbol{A} + \boldsymbol{\mathcal{E}}_m))\boldsymbol{v} - \boldsymbol{V}_m \exp(\tau\boldsymbol{H}_m)\beta\boldsymbol{e}_1 \|, \end{aligned}$$

then bounds on $\tilde{\epsilon}_m$ have been given in the literature of exact Krylov methods [46, 47, 48]. However, our main focus is not so much on the bounds on $\tilde{\epsilon}_m$, but on the *true* error

$$\epsilon_m = \| \exp(\tau \boldsymbol{A})\boldsymbol{v} - \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 \|.$$

The relationship between ϵ_m and $\tilde{\epsilon}_m$ is straightforward from the triangle inequality

$$\epsilon_m \leq \| \exp(\tau \boldsymbol{A})\boldsymbol{v} - \exp(\tau \tilde{\boldsymbol{A}})\boldsymbol{v} \| + \| \exp(\tau \tilde{\boldsymbol{A}})\boldsymbol{v} - \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 \| = \tilde{\epsilon}_m + \delta_m^{err},$$

where we define the *error gap*

$$\delta_m^{\text{err}} = \| \exp(\tau \boldsymbol{A})\boldsymbol{v} - \exp(\tau \tilde{\boldsymbol{A}})\boldsymbol{v} \|. \tag{3.18}$$

With upper bounds on $\tilde{\epsilon}_m$ ready in hand, it remains to get good bounds on δ_m^{err}, which turns out to be a matrix perturbation analysis that we discuss next.

3.4.2 Bounding the error gap

In [51, Chap. 10.2], Higham obtained

$$\exp(\tau \tilde{\boldsymbol{A}}) = \exp(\tau \boldsymbol{A}) + \int_0^\tau \exp((\tau - s)\boldsymbol{A})\boldsymbol{\mathcal{E}}_m \exp(s\boldsymbol{A}) ds + \mathcal{O}\left(\|\tau \boldsymbol{\mathcal{E}}_m\|^2 \right). \tag{3.19}$$

Post-multiplying each side by $\boldsymbol{v}$, we get

$$\exp(\tau \tilde{\boldsymbol{A}})\boldsymbol{v} = \exp(\tau \boldsymbol{A})\boldsymbol{v} + \int_0^\tau \exp((\tau - s)\boldsymbol{A})\boldsymbol{\mathcal{E}}_m \exp(s\boldsymbol{A})\boldsymbol{v} ds + \mathcal{O}\left(\|\tau \boldsymbol{\mathcal{E}}_m\|^2 \right).$$

Hence

$$\delta_m^{\text{err}} \leq \int_0^\tau \| \exp((\tau - s)\boldsymbol{A})\| \|\boldsymbol{\mathcal{E}}_m\| \| \exp(s\boldsymbol{A})\| \|\boldsymbol{v}\| ds + \mathcal{O}(\|\tau \boldsymbol{\mathcal{E}}_m\|^2),$$

which leads directly to the following statement.

Theorem 3.4.1. *For any arbitrary* $\boldsymbol{A}$, $\boldsymbol{v}$ *and* $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{\mathcal{E}}_m$ *from the inexact Krylov subspace method, we have*

$$\delta_m^{err} = \| \exp(\tau \boldsymbol{A})\boldsymbol{v} - \exp(\tau \tilde{\boldsymbol{A}})\boldsymbol{v}\| \leq \beta h_{\boldsymbol{A}}^2 \|\tau \boldsymbol{\mathcal{E}}_m\| + \mathcal{O}(\|\tau \boldsymbol{\mathcal{E}}_m\|^2),$$

where $\beta = \|\boldsymbol{v}\|$ *and* $h_{\boldsymbol{A}} = \max\limits_{s\in[0,\tau]} \| \exp(s\boldsymbol{A})\|$ *is the so-called 'hump' on* $[0, \tau]$.

When $\boldsymbol{A}$ originates from a Markov chain as is the case in the CME, (i.e., with nonnegative off-diagonal elements, negative diagonal elements and zero column sums) it is known that $\| \exp(s\boldsymbol{A})\|_1 = 1, \forall s \geq 0$, (see for example [51, section 2.3]). We can then draw from (3.19) that

$$\| \exp(\tau \boldsymbol{A})\boldsymbol{v} - \exp(\tau \tilde{\boldsymbol{A}})\boldsymbol{v}\|_1 \leq \int_0^\tau \| \exp((\tau - s)\boldsymbol{A})\|_1 \|\boldsymbol{\mathcal{E}}_m\|_1 \| \exp(s\boldsymbol{A})\|_1 \|\boldsymbol{v}\|_1 ds + \mathcal{O}(\|\tau \boldsymbol{\mathcal{E}}_m\|_1^2).$$

and using in addition the fact that a probability vector has $\|\boldsymbol{v}\|_1 = 1$, we get the following simplified result.

Theorem 3.4.2. *Let* $\boldsymbol{A}$ *be the transition rate matrix of a Markov chain, then given a probability vector* $\boldsymbol{v}$ *and the perturbed* $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{\mathcal{E}}_m$ *from the inexact Krylov subspace method, we have*

$$\| \exp(\tau \boldsymbol{A})\boldsymbol{v} - \exp(\tau \tilde{\boldsymbol{A}})\boldsymbol{v}\|_1 \leq \|\tau \boldsymbol{\mathcal{E}}_m\|_1 + \mathcal{O}(\|\tau \boldsymbol{\mathcal{E}}_m\|_1^2).$$

3.4.3 Series expansion of the error

In [47], Saad derived the following series expansion of the error produced by the exact Krylov subspace method

$$\exp(\tau \boldsymbol{A})\boldsymbol{v} - \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 = \tau h_{m+1,m} \sum_{k=1}^{\infty} \boldsymbol{e}_m^T \varphi_k(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 (\tau \boldsymbol{A})^{k-1} \boldsymbol{v}_{m+1}, \qquad (3.20)$$

of which the first term in the series was argued to be a good estimate of the error when the stepsize τ is small enough. Here, the functions φ_k are defined as

$$\varphi_k(x) = \sum_{i=0}^{\infty} \frac{x^i}{(i+k)!}, \quad \varphi_k(0) = \frac{1}{k!},$$

which implies that

$$0 \leq \varphi_k(x) \leq \frac{\varphi_{k-1}(x)}{k} \leq \cdots \leq \frac{e^x}{k!}, \quad \text{if } x \geq 0.$$

In the context of the inexact Krylov subspace method, we can use (3.4) and substitute $\boldsymbol{A}$ for $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{\mathcal{E}}_m$ in (3.20), but that will bring up the unwieldy issue of the error gap again. Instead, we generalize the expansion in the following statement that reveals how the terms involving $\boldsymbol{\mathcal{E}}_m$ break out in a strikingly neat way.

Theorem 3.4.3. *The (true) error in the inexact Krylov subspace method for the matrix exponential has the series expansion*

$$\begin{aligned} &\exp(\tau \boldsymbol{A})\boldsymbol{v} - \boldsymbol{V}_m \exp(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 \\ &\quad = \sum_{k=1}^{\infty} (\tau \boldsymbol{A})^{k-1} \left[\tau h_{m+1,m} \left(\boldsymbol{e}_m^T \varphi_k(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 \right) \boldsymbol{v}_{m+1} - \tau \boldsymbol{\mathcal{E}}_m \boldsymbol{V}_m \varphi_k(\tau \boldsymbol{H}_m)\beta \boldsymbol{e}_1 \right]. \end{aligned} \tag{3.21}$$

Proof. As in the proof of [47, Theorem 5.1], define the error in approximating $\varphi_k(\boldsymbol{A})\boldsymbol{v}$ by projecting it onto $\boldsymbol{V}_m$ as $\boldsymbol{s}_m^k = \varphi_k(\boldsymbol{A})\boldsymbol{v} - \boldsymbol{V}_m \varphi_k(\boldsymbol{H}_m)\beta \boldsymbol{e}_1$.

From the definition of φ_k and the fact that $\varphi_k(\mathbf{0})\boldsymbol{v} = \boldsymbol{V}_m\varphi_k(\mathbf{0})\beta\boldsymbol{e}_1$, we directly have

$$\begin{aligned}
\varphi_k(\boldsymbol{A})\boldsymbol{v} &= \boldsymbol{A}\varphi_{k+1}(\boldsymbol{A})\boldsymbol{v} + \varphi_k(\mathbf{0})\boldsymbol{v} \\
&= \boldsymbol{A}[\boldsymbol{V}_m\varphi_{k+1}(\boldsymbol{H}_m)\beta\boldsymbol{e}_1 + \boldsymbol{s}_m^{k+1}] + \varphi_k(\mathbf{0})\boldsymbol{v} \\
&= \boldsymbol{V}_m[\varphi_k(\mathbf{0})\beta\boldsymbol{e}_1 + \boldsymbol{H}_m\varphi_{k+1}(\boldsymbol{H}_m)\beta\boldsymbol{e}_1] + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T\varphi_{k+1}(\boldsymbol{H}_m)\beta\boldsymbol{e}_1 \\
&\quad - \boldsymbol{\mathcal{E}}_m\boldsymbol{V}_m\varphi_{k+1}(\boldsymbol{H}_m)\beta\boldsymbol{e}_1 + \boldsymbol{A}\boldsymbol{s}_m^{k+1} \\
&= \boldsymbol{V}_m\varphi_k(\boldsymbol{H}_m)\beta\boldsymbol{e}_1 + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T\varphi_{k+1}(\boldsymbol{H}_m)\beta\boldsymbol{e}_1 \\
&\quad - \boldsymbol{\mathcal{E}}_m\boldsymbol{V}_m\varphi_{k+1}(\boldsymbol{H}_m)\beta\boldsymbol{e}_1 + \boldsymbol{A}\boldsymbol{s}_m^{k+1},
\end{aligned}$$

resulting in another expression for $\boldsymbol{s}_m^k$ through a recurrence

$$\boldsymbol{s}_m^k = h_{m+1,m}\left(\boldsymbol{e}_m^T\varphi_{k+1}(\boldsymbol{H}_m)\beta\boldsymbol{e}_1\right)\boldsymbol{v}_{m+1} - \boldsymbol{\mathcal{E}}_m\boldsymbol{V}_m\varphi_{k+1}(\boldsymbol{H}_m)\beta\boldsymbol{e}_1 + \boldsymbol{A}\boldsymbol{s}_m^{k+1}.$$

Using these expressions of the error terms gives

$$\begin{aligned}
&\exp(\boldsymbol{A})\boldsymbol{v} - \boldsymbol{V}_m\exp(\boldsymbol{H}_m)\beta\boldsymbol{e}_1 = \boldsymbol{s}_m^0 \\
&= h_{m+1,m}\left(\boldsymbol{e}_m^T\varphi_1(\boldsymbol{H}_m)\beta\boldsymbol{e}_1\right)\boldsymbol{v}_{m+1} - \boldsymbol{\mathcal{E}}_m\boldsymbol{V}_m\varphi_1(\boldsymbol{H}_m)\beta\boldsymbol{e}_1 + \boldsymbol{A}\boldsymbol{s}_m^1 \quad = \ldots \\
&= h_{m+1,m}\sum_{k=1}^{j}\left(\boldsymbol{e}_m^T\varphi_k(\boldsymbol{H}_m)\beta\boldsymbol{e}_1\right)\boldsymbol{A}^{k-1}\boldsymbol{v}_{m+1} - \sum_{k=1}^{j}\boldsymbol{A}^{k-1}\boldsymbol{\mathcal{E}}_m\boldsymbol{V}_m\varphi_k(\boldsymbol{H}_m)\beta\boldsymbol{e}_1 + \boldsymbol{A}^j\boldsymbol{s}_m^j,
\end{aligned}$$

in which

$$\|\boldsymbol{A}^j\boldsymbol{s}_m^j\| \le \|\boldsymbol{A}\|^j\|\boldsymbol{s}_m^j\| \le \|\boldsymbol{A}\|^j\beta\left(\varphi_j(\|\boldsymbol{A}\|) + \varphi_j(\|\boldsymbol{H}_m\|)\right) \le \beta\left(e^{\|\boldsymbol{A}\|} + e^{\|\boldsymbol{H}_m\|}\right)\frac{\|\boldsymbol{A}\|^j}{j!}$$

converges to 0 as $j \to \infty$. Taking this into account in the sums above, we get

$$\begin{aligned}
&\exp(\boldsymbol{A})\boldsymbol{v} - \boldsymbol{V}_m\exp(\boldsymbol{H}_m)\beta\boldsymbol{e}_1 \\
&\quad = \sum_{k=1}^{\infty}\boldsymbol{A}^{k-1}[h_{m+1,m}\left(\boldsymbol{e}_m^T\varphi_k(\boldsymbol{H}_m)\beta\boldsymbol{e}_1\right)\boldsymbol{v}_{m+1} - \boldsymbol{\mathcal{E}}_m\boldsymbol{V}_m\varphi_k(\boldsymbol{H}_m)\beta\boldsymbol{e}_1].
\end{aligned}$$

Finally, if we rescale the inexact Arnoldi relation in (3.4) with the stepsize τ, we get (3.21). □

3.4.4 Exactness in the case of truncated approximations

The analysis so far has not made any assumption on the structure of $\boldsymbol{A}$, $\boldsymbol{v}$ or $\boldsymbol{E}_k$. In this section, we show a counter-intuitive result that, when $\boldsymbol{E}_k$ arises from a truncated approximation of a special form of the matrix $\boldsymbol{A}$, the inexact scheme can be exact.

Consider a banded matrix, and assume that $\boldsymbol{v} = \boldsymbol{e}_1 = (1, 0, \ldots, 0)^T$. The multiplication of $\boldsymbol{A}$ with $\boldsymbol{v}$ will therefore not involve the contribution of elements located in the trailing submatrices of $\boldsymbol{A}$. Generalizing this observation, we get the following result that arises frequently in the CME discussed in section 3.2 and is therefore of wide interest.

Theorem 3.4.4. *Let $l \geq 0$, $k - l \geq 2$, assume that*

$$\boldsymbol{A} = \left(\begin{array}{c|c} \boldsymbol{A}_k & \boldsymbol{B} \\ \hline \boldsymbol{C} & \boldsymbol{D} \end{array}\right) \in \mathbb{R}^{n \times n},$$

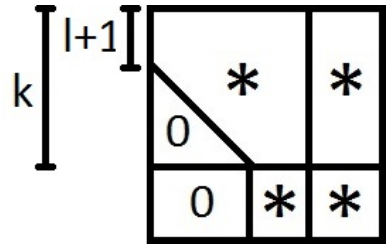
where

$$\boldsymbol{A}_k \in \mathbb{R}^{k \times k}, \quad (\boldsymbol{A}_k)_{ij} = 0 \text{ if } i > j + l,$$

and

$$\boldsymbol{C} \in \mathbb{R}^{(n-k) \times k}, \quad (\boldsymbol{C})_{ij} = 0 \text{ if } j \leq k - l - 1.$$

That is, $\boldsymbol{A}$ visually has the form:



Also assume that $\boldsymbol{v} = \boldsymbol{e}_1$ *and the relaxation matrices are identical,*

$$\boldsymbol{E} = \left(\begin{array}{c|c} \mathbf{0} & -\boldsymbol{B} \\ \hline -\boldsymbol{C} & -\boldsymbol{D} \end{array}\right) \in \mathbb{R}^{n\times n}.$$

Then if $m \leq M = \max\{j : (j-1)l + 1 \leq k - l - 1\}$*, where* m *is the dimension of the basis built by the inexact Arnoldi algorithm based on matrix-vector products with* $\boldsymbol{A} + \boldsymbol{E}$*, we have*

$$\boldsymbol{\mathcal{E}}_m = \mathbf{0},$$

so that

$$\exp(\tau \boldsymbol{A}) = \exp(\tau \tilde{\boldsymbol{A}}),$$

and

$$\delta_m^{err} = \mathbf{0}.$$

Proof. Observe first that, because of the special forms of $\boldsymbol{A}$ and $\boldsymbol{v}$, the $\boldsymbol{v}_j$ produced by the Arnoldi process is such that

$$(\boldsymbol{v}_j)_i = 0, \quad i > (j-1)l + 1, \quad 1 \leq j \leq M.$$

This means that, up to $j = M$, the exact Arnoldi process (where multiplications are performed with $\boldsymbol{A}$) and the inexact Arnoldi process (where they are instead performed with $\boldsymbol{A} + \boldsymbol{E}$) coincide, due to the fact that the multiplications only depend on the first $k - l - 1$ columns of $\boldsymbol{A}$ and $\boldsymbol{A} + \boldsymbol{E}$, which are the same.

Because the first $k - l - 1$ columns of $\boldsymbol{E}$ are zero, and only the first $(j-1)l + 1 \leq k - l - 1$ elements of the vectors $\boldsymbol{v}_j$ are nonzero for $1 \leq j \leq m \leq M$, we have

$\boldsymbol{E}\boldsymbol{v}_j = \mathbf{0}$, and therefore

$$\boldsymbol{\mathcal{E}}_m = \sum_{j=1}^{m} \boldsymbol{E}\boldsymbol{v}_j \boldsymbol{v}_j^T = \mathbf{0}.$$

Hence $\boldsymbol{A} = \tilde{\boldsymbol{A}}$ and naturally $\delta_m^{\text{err}} = \| \exp(\tau \boldsymbol{A})\boldsymbol{v} - \exp(\tau \tilde{\boldsymbol{A}})\boldsymbol{v}\| = 0$. □

Remark 3.4.5. The analysis in Theorem 3.4.4 will explain results apparently intriguing in the experiments, as the reader will soon discover in the following section. Note however that it only works for $\boldsymbol{v} = \boldsymbol{e}_1$. Since the step-by-step integration scheme (3.11) is typically used, $\boldsymbol{v}$ will not remain $\boldsymbol{e}_1$ past the first step, in which case the analysis of δ_m^{err} done in the previous section takes effect.

Remark 3.4.6. Even though this theorem shows that we have $\boldsymbol{\mathcal{E}}_m = 0$ and $\exp(\tau \boldsymbol{A}) = \exp(\tau \tilde{\boldsymbol{A}}), \forall \tau$, with a Krylov subspace of small dimension, this does not mean that the inexact Krylov approximation has no error. As shown in Theorem 3.4.3, even if $\boldsymbol{\mathcal{E}}_m = 0$, the expansion collapses to (3.20), which is generally not zero.

3.5 Numerical examples

We illustrate some of the theoretical results using three examples. The first two examples illustrate how Proposition 3.3.1 can be applied, by first showing how the relaxation scheme works according to the proposition, and secondly how the scheme fails when the condition of the proposition is not satisfied. The third example is a special one to demonstrate the peculiar behavior hinted in Remark 3.4.5: the scheme still works well even though the (sufficient) condition of Proposition 3.3.1 is not satisfied due to the reason given in Theorem 3.4.4. The examples were implemented in MATLAB.

We recall the following quantities that were given in the text to assess the inexact

method:

$$\text{true residual} = \|\boldsymbol{r}_m\| = \|\boldsymbol{V}_m\boldsymbol{H}_m\boldsymbol{y}_m - \boldsymbol{A}\boldsymbol{V}_m\boldsymbol{y}_m\|$$

$$\text{computed residual} = \|\tilde{\boldsymbol{r}}_m\| = |h_{m+1,m}\boldsymbol{e}_m^T\boldsymbol{y}_m|$$

$$\text{residual gap} = \delta_m^{\text{res}} = \|\boldsymbol{r}_m - \tilde{\boldsymbol{r}}_m\| = \|\boldsymbol{\mathcal{E}}_m\boldsymbol{V}_m\boldsymbol{y}_m\|$$

$$\text{true error} = \epsilon_m = \|\exp(\tau\boldsymbol{A})\boldsymbol{p}_0 - \boldsymbol{V}_m\boldsymbol{y}_m\|,$$

where $\boldsymbol{y}_m = \exp(\tau\boldsymbol{H}_m)\beta\boldsymbol{e}_1$, with a given τ, and with $\boldsymbol{H}_m$ and $\boldsymbol{V}_m$ constructed by the inexact Krylov method using an initial given vector $\boldsymbol{v}$ with $\beta = \|\boldsymbol{v}\|$.

3.5.1 Example 1 - Illustration of the residual gap when Proposition 3.1 is satisfied

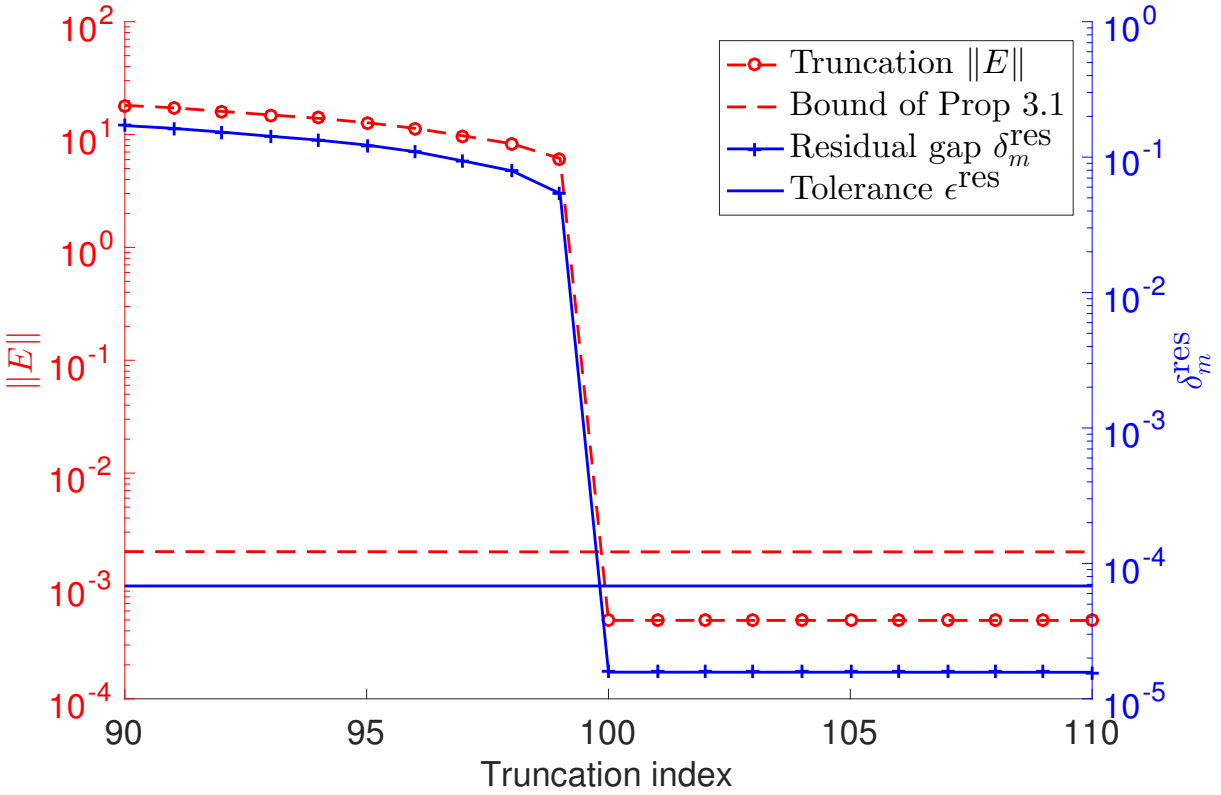


Figure 3.1: Example 1: when the principal 100×100 submatrix becomes contained in the truncated matrix, the bound condition of Proposition 3.3.1 is satisfied, resulting in δ_m^{res} within the tolerance.

We take a 1000×1000 matrix, where the principal 100×100 submatrix is uniformly distributed in $[0, 1]$, and entries outside this submatrix are uniformly distributed in $[0, 10^{-6}]$. The initial vector is $\boldsymbol{v} = (10^{-3}, \ldots, 10^{-3})^T$. We take the time point $\tau = 10^{-3}$, and tolerance on the residual gap $\epsilon^{res} = 10^{-3}$. The Krylov subspace is chosen to be of dimension $m = 15$.

For the inexact scheme, we define $\boldsymbol{A}_k$ to be a 1000×1000 matrix containing the principal $k \times k$ submatrix of $\boldsymbol{A}$ and 0 outside. Inexact multiplications with $\boldsymbol{A}$ are then performed with $\boldsymbol{A}_k$ instead. Observations are shown in Fig. 3.1. Since the key entries of $\boldsymbol{A}$ are in the principal 100×100 submatrix, it s clear that if $k < 100$, then $\boldsymbol{A}_k$ will leave out vital entries and inflate the error matrix $\boldsymbol{E} = \boldsymbol{A} - \boldsymbol{A}_k$, which in turn will cause the residual gap to be large. However, if $k \geq 100$, then $\|\boldsymbol{E}\| \leq \frac{\epsilon^{res}}{m\beta \| \exp(\tau \boldsymbol{H}_m)\|}$, which is the bound condition in Proposition 3.3.1, and therefore $\delta_m^{\text{res}} \leq \epsilon^{res}$ as guaranteed there. Fig. 3.1 illustrates this with the truncation index $90 \leq k \leq 110$.

3.5.2 Example 2 - Illustration of the residual gap when Proposition 3.1 is not satisfied

We now consider another 1000×1000 matrix, where the main diagonal is uniformly distributed in $[0, 1]$, and the off-diagonal entries are uniformly distributed in $[0, 10^{-6}]$. As in the first example, we keep $\boldsymbol{v} = (10^{-3}, \ldots, 10^{-3})^T$, $\tau = 10^{-3}$, $\epsilon^{res} = 10^{-3}$, and $m = 15$. The inexact setup is also the same, with inexact matrix-vector products against $\boldsymbol{A}$ performed using $\boldsymbol{A}_k$, where $\boldsymbol{A}_k$ contains the $k \times k$ principal submatrix of $\boldsymbol{A}$ and zeros elsewhere.

As Fig. 3.2 shows, in this example, the residual gap δ_m^{res} is consistently above the tolerance on the residual gap ϵ^{res} even when k is almost the size of $\boldsymbol{A}$. The reason for this is that since the diagonal elements are significant to be omitted, truncating even only one or two of them would make $\|\boldsymbol{E}\|$ large and not satisfy the bound condition of Proposition 3.3.1.

The conclusion to draw from these two examples is that when inexactness is achieved in a reasonable way, the inexact Krylov method works well. Depending on the

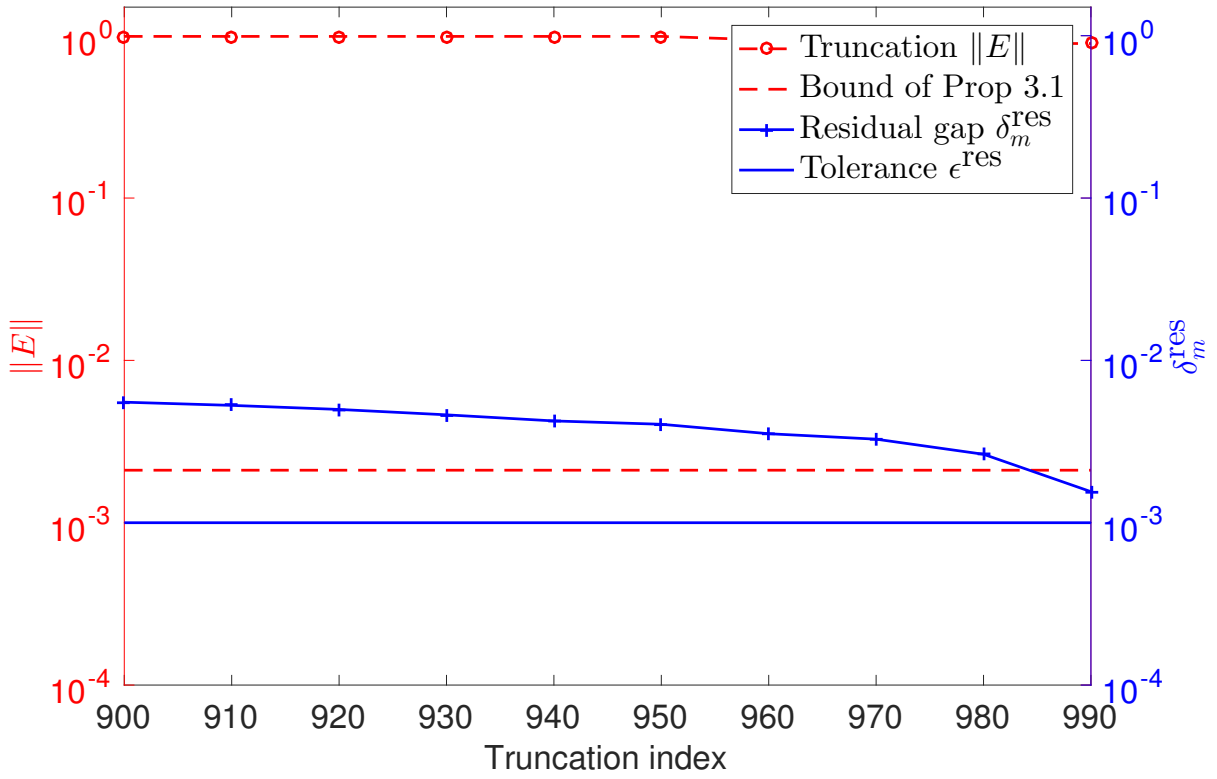


Figure 3.2: Example 2: The bound condition of Proposition 3.3.1 is never satisfied, and δ_m^{res} is greater than the tolerance.

particular problem at hand, one can decide how to relax the matrix-vector multiplications to satisfy Proposition 3.3.1, in which case the residual gap can be controlled by ϵ^{res}, ensuring that the computed residual $\|\tilde{\boldsymbol{r}}_m\|$ serves as a reliable approximation of the true residual $\|\boldsymbol{r}_m\|$.

3.5.3 Example 3 - Illustration of the error and residuals when Theorem 4.4 is satisfied

We consider the CME arising from the Michaelis-Menten enzyme kinetics, which is a well known system of biochemical reactions in cell biology. There are four species: substrates (S), enzymes (E), enzyme-substrate complexes (ES) and products (P), interacting according to the three chemical reactions listed in Table 3.1, and we took reaction rates as in [54]. The state vector is $\boldsymbol{x} = ([\text{P}], [\text{E}], [\text{S}], [\text{ES}])^T$, where [X] is the

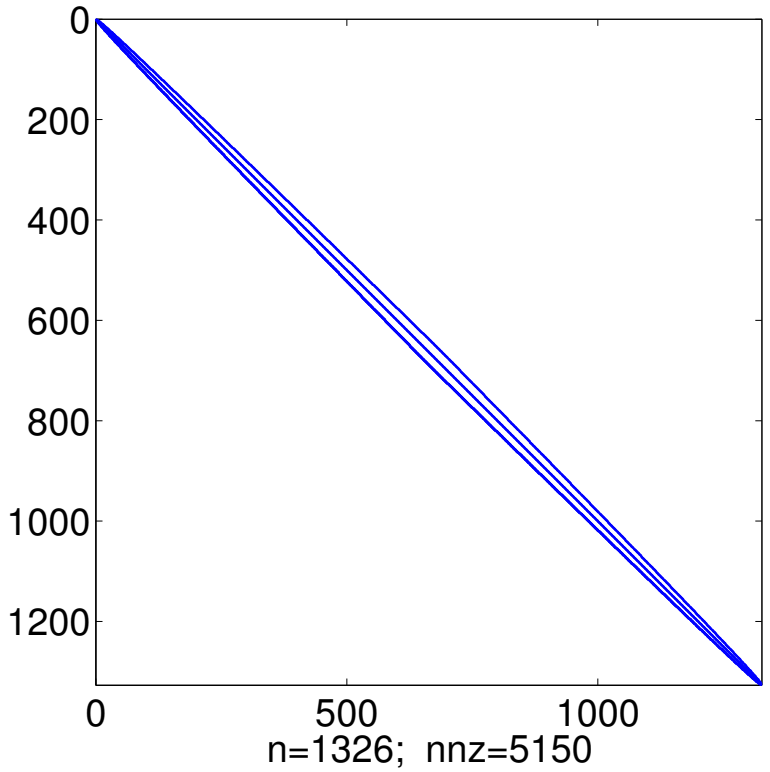


Figure 3.3: Sparsity pattern of the matrix $\boldsymbol{A}$ from the CME of the Michaelis-Menten enzyme kinetics in Example 3.

current number of copies of species X. If we start with $\boldsymbol{x}_1 = (0, 50, 50, 0)^T$, i.e., a maximum of 50 substrates, the resulting matrix $\boldsymbol{A}$ of the underlying CME is of dimension $n = 1{,}326$. We use MATLAB's `expm` command to check for correctness. Fig. 3.3 shows the sparsity pattern of $\boldsymbol{A}$.

Table 3.1: Michaelis-Menten reactions and propensities.

	reaction	propensity	rate constant (s^{-1})
1.	$\mathrm{E} + \mathrm{S} \xrightarrow{\kappa_1} \mathrm{ES}$	$\alpha_1 = \kappa_1\ [\mathrm{E}]\ [\mathrm{S}]$	$\kappa_1 = 1.0$
2.	$\mathrm{ES} \xrightarrow{\kappa_2} \mathrm{E} + \mathrm{S}$	$\alpha_2 = \kappa_2\ [\mathrm{ES}]$	$\kappa_2 = 1.0$
3.	$\mathrm{ES} \xrightarrow{\kappa_3} \mathrm{E} + \mathrm{P}$	$\alpha_3 = \kappa_3\ [\mathrm{ES}]$	$\kappa_3 = 0.1$

Since we know that the system starts in state $\boldsymbol{x}_1$, the initial probability vector is $\boldsymbol{p}_0 = \boldsymbol{e}_1$, and in the spirit of (3.6), we compute the approximation

$$\exp(\tau \boldsymbol{A})\boldsymbol{p}_0 \approx \exp(\tau \boldsymbol{A}_J)\boldsymbol{p}_0,$$

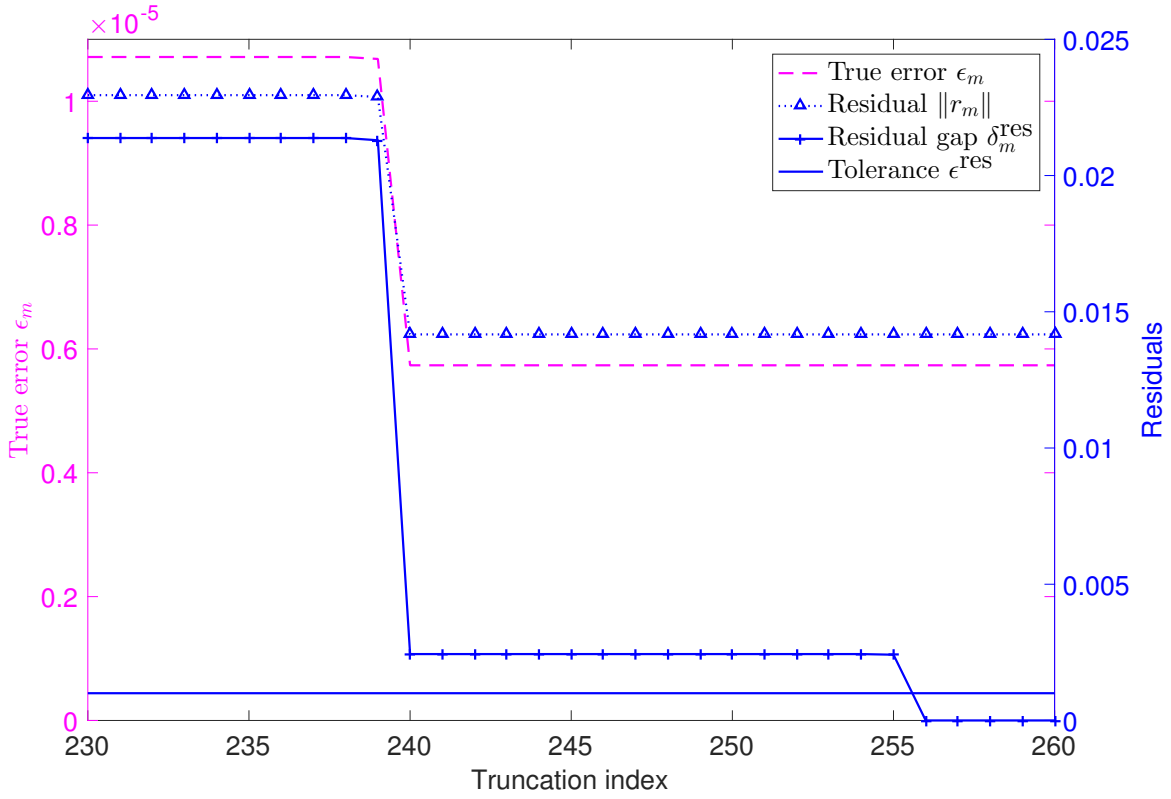


Figure 3.4: Example 3: The true error (on the left y-axis), and the true residual and residual gap (on the right y-axis)

where $\boldsymbol{A}_J$ is a padded truncation of $\boldsymbol{A}$. We simply take a principal submatrix instead of the more general scheme where J can be an arbitrary subset of $\{1, \ldots, n\}$. There is no loss of generality because it can be assumed that there has been a reordering $\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}$ where $\boldsymbol{P}$ is an appropriate permutation matrix. Our aim is to test the theory developed here and not really to craft efficient implementation details with elaborate sparse data structures that are best communicated elsewhere. We vary $|J|$, the cardinality of J that determines the size of the truncation from 230 to 280, so that $|J|$ ranges from about 17% to 21% of n in this example. We take $\boldsymbol{v} = \boldsymbol{e}_1$, $\tau = 10^{-2}$, and $m = 30$.

Figs. 3.4 and 3.5 show the results. We see on the first plot a decrease of $\|\boldsymbol{\mathcal{E}}_m\|$ as the truncation size $|J|$ increases. This example has the particularity of illustrating the phenomenon described in §3.4.4, where $\boldsymbol{\mathcal{E}_m}$ becomes a zero matrix for a big enough truncation size $|J|$, even though $|J|$ is still only about 20% of n. Because of this

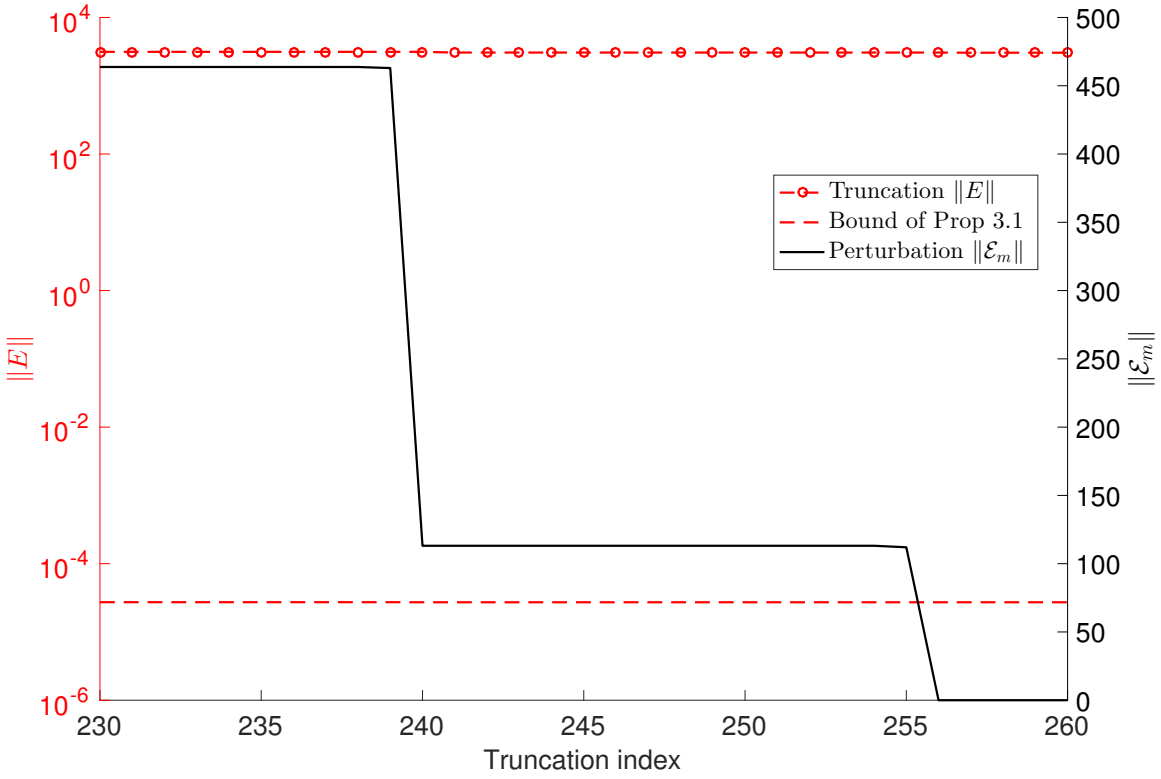


Figure 3.5: Example 3: $\|\boldsymbol{E}\|$ and $\|\boldsymbol{\mathcal{E}_m}\|$; computed as $|J|$ increases from 230 to 280; $\boldsymbol{v} = \boldsymbol{e}_1$, $\tau = 10^{-2}$, and $m = 30$

phenomenon, the residual gap becomes 0, as the left plot shows. These results agree with the theory presented here. Note also on the first plot how the stair functions change in unison, illustrating the connection between the residuals and the error.

3.6 Conclusion

In this chapter, we have analyzed inexact Krylov methods for approximating the action of the matrix exponential and provided insights into why they can be successful. We obtained results that in hindsight connect well with previous results, but it is worth recalling that exactly how such inexact methods related to previous works was unclear at the beginning. The rigorous treatment performed in this work made the connection clear and established the details. This therefore fills a gap in the literature. We also brought into focus a particularly attractive aspect of inexact methods: they set a framework that

encompasses model reduction methods in a generic way. We gave the important application of solving the CME as an example to motivate this viewpoint, with truncation methods such as the FSP method that fit naturally in the framework.

CHAPTER 4

CHEMICAL MASTER EQUATION WITH TIME-VARYING RATES

4.1 Introduction

So far we have considered two different approaches of solving the CME. It can either be done indirectly with Monte Carlo methods, such as the SSA, or directly solved with the FSP and its variants, some of which have been mentioned in Chapter 2. These earlier FSP algorithms only work if the reaction rates in the biological system are constant. As detailed in Chapter 2, the CME then becomes

$$\dot{\boldsymbol{p}}(t) = \boldsymbol{A} \cdot \boldsymbol{p}(t), \tag{4.1}$$

where $\boldsymbol{p}(t)$ stores the probabilities of the state space at time t, and $\boldsymbol{A}$ stores the propensities at which the system travels from one state to another within an infinitesimal time period. Note that entries in $\boldsymbol{A}$ only depend on the reaction rates, therefore it is time-independent if the reaction rates are constant throughout the time period of interest.

The theoretical solution for (4.1) is

$$\boldsymbol{p}(t_f) = \exp(t_f \boldsymbol{A})\boldsymbol{p}(0), \tag{4.2}$$

so the variable time-stepping FSP algorithms integrate from initial time t_0 to the end time of interest t_f with the basic formula

$$\boldsymbol{p}(t_k + h_k) \approx \exp(h_k \boldsymbol{A})\boldsymbol{p}(t_k). \tag{4.3}$$

However, if the reaction rates in the biological system change over time, then $\boldsymbol{A}$

becomes time-dependent, and the CME changes to

$$\dot{\boldsymbol{p}}(t) = \boldsymbol{A}(t) \cdot \boldsymbol{p}(t), \tag{4.4}$$

the solution of which is no longer (4.2), implying the current FSP integration formula (4.3) is not exact. In this case, if we want to directly solve the CME for the probability distributions, a general ODE solver, such as Adams-Bashforth, Runge-Kutta, or Backward-differentiation formula, must be employed to solve (4.4).

We will develop a new integration method for (4.4) based on the Magnus expansion in chapter 5. Here, we first review several different scenarios in which problems in the form of (4.4) can arise in biologically relevant contexts. The chapter is organized as follows: Section 4.2 examines several biological problems in which the time-dependence of the reaction rates arises naturally. In Section 4.3, we reduce non-homogeneous ODE problems into (4.4). Finally, we discuss the delay CME in section 4.4 and analyze how it can be transformed into a CME with time-varying rates.

4.2 Biological origins of CME with time-varying rates

There are various reasons for which the reaction rates in a biological system can change over time in the biological context. For instance, in [55], the authors modeled the competition between several T cell clonotypes via coupled birth-death processes where the size of one clonotype can affect the birth rates of the others. They also observed that in human, all T cell clonotypes ultimately became extinct, and this phenomenon is incorporated into the model as time-dependent birth rates for all clonotypes. As the birth rates decrease to 0 over time, all T cell clonotypes eventually disappear. Hence, in this case, the time-dependence of the reaction rates originates from biological observations.

An SIR model is an epidemiological model that computes the theoretical number of people infected with a contagious illness over time [56, 57, 58, 59]. In the model, the susceptible population (S) can become ill from contact with the infected population (I).

Eventually, they recover (R) and might become susceptible to the disease again. For some diseases, the contact rate between the infected population and the susceptible population appears to be periodic [56, 59]. This results in oscillatory reaction rates.

A transcriptional regulatory system in an E. coli cell is modeled in [54]. It contains DNA templates, which can transcribe RNA. The RNAs in turn translate protein monomers. Two monomers can combine into one dimer, and this dimer can bind to the DNA template to start or stop the RNA transcription process. The reaction of monomers combining to one dimer and the reaction of the dimer binding to the DNA are both time-dependent, because as the cell grows, its volume increases, which decreases the probability of the reactants colliding in order for the reactions to occur.

In chapter 5, all three problems described above will be solved with a Monte Carlo method and traditional ODE solvers, as well as a Magnus-expansion-based integration method that has not been investigated for solving such biological problems in the literature. The models will therefore be described in detail in those numerical tests. Many more models with time-varying reaction rates exist in the literature.

4.3 Non-homogeneous equations

In [60], the authors described a technique to transform a non-autonomous and non-homogeneous linear differential equation set into (4.4). Let the ODE problem be in the form of

$$\frac{d^N}{dt^N}\boldsymbol{x}(t) + \boldsymbol{f}_{N-1}(t) \cdot \frac{d^{N-1}}{dt^{N-1}}\boldsymbol{x}(t) + \cdots + \boldsymbol{f}_1(t) \cdot \frac{d}{dt}\boldsymbol{x}(t) + \boldsymbol{f}_0(t)\boldsymbol{x}(t) = \boldsymbol{g}(t), \qquad (4.5)$$

where $\boldsymbol{x}(t), \boldsymbol{g}(t) \in \mathbb{C}^{m\times d}$ and $\boldsymbol{f}_i(t) \in \mathbb{C}^{m\times m}$. By introducing a new variable

$$\boldsymbol{z}(t) = \begin{pmatrix} \boldsymbol{x}(t) \\ \frac{d}{dt}\boldsymbol{x}(t) \\ \vdots \\ \frac{d^{N-1}}{dt^{N-1}}\boldsymbol{x}(t) \\ \mathbf{1} \end{pmatrix} \in \mathbb{C}^{(N+1)m\times d},$$

the equation (4.5) can be rewritten as

$$\frac{d}{dt}\boldsymbol{z}(t) = \boldsymbol{M}(t)\cdot\boldsymbol{z}(t), \tag{4.6}$$

where the matrix $\boldsymbol{M}(t)$ is defined as

$$\boldsymbol{M}(t) = \begin{pmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ -\boldsymbol{f}_0(t) & -\boldsymbol{f}_1(t) & -\boldsymbol{f}_2(t) & \cdots & -\boldsymbol{f}_{N-2}(t) & -\boldsymbol{f}_{N-1}(t) & \boldsymbol{g}(t) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \in \mathbb{C}^{(N+1)m\times(N+1)m}.$$

Note that the equation (4.6) is in the form of (4.4) and therefore can be solved using the numerical integration methods to be discussed in the next chapter.

4.4 Delay CME

In [61], the authors formulated the delay CME (DCME), in which some reactions require a delay time period to finish. In particular, they considered a protein production

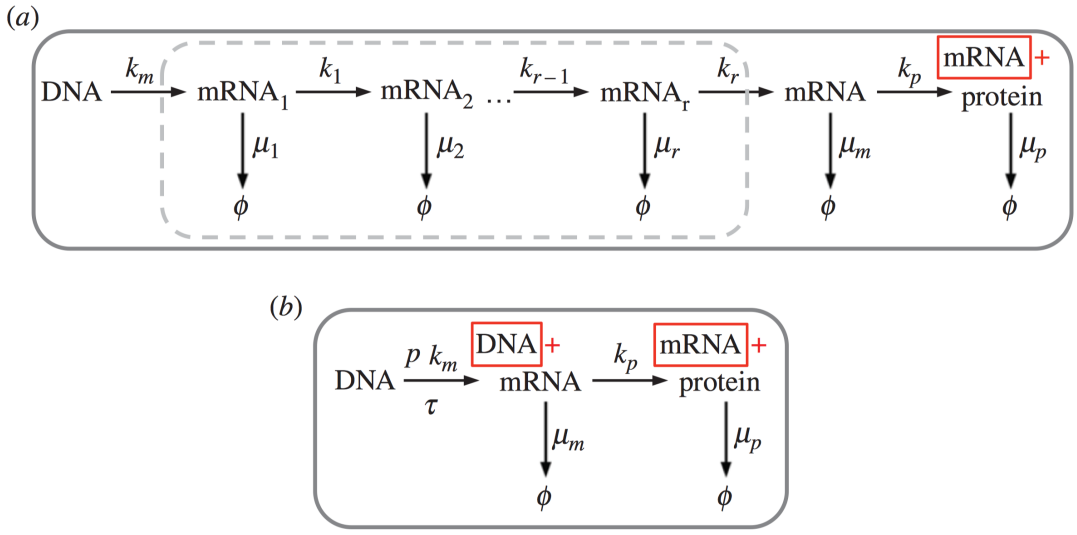


Figure 4.1: (a) Full-sized model. (b) Delay model

model in which the messenger RNA (mRNA) requires r intermediate steps to mature:

$$\text{DNA} \xrightarrow{k_m} \text{DNA} + \text{mRNA}_1, \tag{4.7}$$

$$\text{mRNA}_1 \xrightarrow{k_1} \text{mRNA}_2, \quad \text{mRNA}_1 \xrightarrow{\mu_1} \varnothing, \tag{4.8}$$

$$\text{mRNA}_2 \xrightarrow{k_2} \text{mRNA}_3, \quad \text{mRNA}_2 \xrightarrow{\mu_2} \varnothing, \tag{4.9}$$

$$\ldots \tag{4.10}$$

$$\text{mRNA}_r \xrightarrow{k_r} \text{mRNA}, \quad \text{mRNA}_r \xrightarrow{\mu_r} \varnothing, \tag{4.11}$$

$$\text{mRNA} \xrightarrow{k_p} \text{mRNA} + \text{protein}, \quad \text{mRNA} \xrightarrow{\mu_m} \varnothing,$$

$$\text{protein} \xrightarrow{\mu_p} \varnothing$$

This model is described in Figure 4.1-a. They then combined those intermediate

reactions (4.7)-(4.11) into one:

$$\begin{aligned} R_1: &\quad \text{DNA} \xrightarrow[\tau]{k'_m} \text{DNA} + \text{mRNA}, \\ R_2: &\quad \text{mRNA} \xrightarrow{k_p} \text{mRNA} + \text{protein}, \\ R_3: &\quad \text{mRNA} \xrightarrow{\mu_m} \varnothing, \\ R_4: &\quad \text{protein} \xrightarrow{\mu_p} \varnothing \end{aligned} \tag{4.12}$$

This model is described in Figure 4.1-b. The intermediate reactions do not occur simultaneously, so the reaction (4.12) in the abridged model is delayed with a random time period τ and reaction rate

$$k'_m = k_m \cdot \frac{\prod_{i=1}^{r} k_i}{\prod_{i=1}^{r}(k_i + \mu_i)},$$

accounting for the mRNA being degraded during the maturation process. Assuming that $\mu = \mu_1 = \cdots = \mu_r$ and $k = k_1 = \cdots = k_r$, the authors found the probability density function (PDF) of τ to be

$$f(t) = \frac{\beta^r t^{r-1}}{(r-1)!} e^{-\beta t} \tag{4.13}$$

and its cumulative distribution function (CDF) is

$$F(t) = 1 - e^{-\beta t} \sum_{k=0}^{r-1} \frac{\beta^{r-1-k}}{(r-1-k)!} t^{r-1-k} \tag{4.14}$$

with $\beta = \mu + k$. The corresponding DCME for state $\boldsymbol{x}$ is

$$\begin{aligned}\frac{\partial}{\partial t}P(\boldsymbol{x},t) = &-\left(\alpha_2(P(\boldsymbol{x},t)) + \alpha_3(P(\boldsymbol{x},t)) + \alpha_4(P(\boldsymbol{x},t))\right)\cdot P(\boldsymbol{x},t)\\ &+ \alpha_2(P(\boldsymbol{x}-\boldsymbol{\nu}_2,t))\cdot P(\boldsymbol{x}-\boldsymbol{\nu}_2,t)\\ &+ \alpha_3(P(\boldsymbol{x}-\boldsymbol{\nu}_3,t))\cdot P(\boldsymbol{x}-\boldsymbol{\nu}_3,t)\\ &+ \alpha_4(P(\boldsymbol{x}-\boldsymbol{\nu}_4,t))\cdot P(\boldsymbol{x}-\boldsymbol{\nu}_4,t)\\ &- \sum_{\boldsymbol{x}_i\in I(\boldsymbol{x})}\int_0^t k'_m\cdot f(\tau)\cdot P(\boldsymbol{x},t;\boldsymbol{x}_i,t-\tau)d\tau\\ &+ \sum_{\boldsymbol{x}_i\in I(\boldsymbol{x})}\int_0^t k'_m\cdot f(\tau)\cdot P(\boldsymbol{x}-\boldsymbol{\nu}_1,t;\boldsymbol{x}_i,t-\tau)d\tau\end{aligned}$$

The first four terms on the right hand side correspond to the non-delay reactions R_2, R_3 and R_4, which, upon occurring with propensities

$$\begin{aligned}\alpha_2 &= k_p\cdot[\text{mRNA}],\\ \alpha_3 &= \mu_m\cdot[\text{mRNA}],\\ \alpha_4 &= \mu_p\cdot[\text{protein}],\end{aligned}$$

instantaneously update the state vector according to their corresponding stoichiometric vectors $\boldsymbol{\nu}_2$, $\boldsymbol{\nu}_3$ and $\boldsymbol{\nu}_4$. The final two terms in the DCME correspond to the delay reaction R_1, accounting for all states in $I(\boldsymbol{x})$, which consists of all possible states that the system is able to follow via a chain of reactions before time t. In general, the joint probability distribution $P(\boldsymbol{x},t;\boldsymbol{x}',t-\tau)$ is not given, and further assumptions are required to simplify the DCME. However, for this specific model, this is not necessary, as the triggering of the delay reaction does not depend on the state at triggering time or the occurrences of other

reactions, hence

$$\begin{aligned}\sum_{\boldsymbol{x}_i \in I(\boldsymbol{x})} \int_0^t k'_m \cdot f(\tau) \cdot P(\boldsymbol{x}, t; \boldsymbol{x}_i, t-\tau) d\tau &= \int_0^t k'_m \cdot f(\tau) \cdot \sum_{\boldsymbol{x}_i \in I(\boldsymbol{x})} P(\boldsymbol{x}, t; \boldsymbol{x}_i, t-\tau) d\tau \\ &= \int_0^t k'_m \cdot f(\tau) \cdot P(\boldsymbol{x}, t) d\tau \\ &= k'_m \cdot F(t) \cdot P(\boldsymbol{x}, t).\end{aligned}$$

Similarly,

$$\sum_{\boldsymbol{x}_i \in I(\boldsymbol{x})} \int_0^t k'_m \cdot f(\tau) \cdot P(\boldsymbol{x} - \boldsymbol{\nu}_1, t; \boldsymbol{x}_i, t-\tau) d\tau = k'_m \cdot F(t) \cdot P(\boldsymbol{x} - \boldsymbol{\nu}_1, t),$$

and the DCME becomes

$$\begin{aligned}\frac{\partial}{\partial t} P(\boldsymbol{x}, t) = &- \left(\alpha_2(P(\boldsymbol{x}, t)) + \alpha_3(P(\boldsymbol{x}, t)) + \alpha_4(P(\boldsymbol{x}, t))\right) \cdot P(\boldsymbol{x}, t) \\ &+ \alpha_2(P(\boldsymbol{x} - \boldsymbol{\nu}_2, t)) \cdot P(\boldsymbol{x} - \boldsymbol{\nu}_2, t) \\ &+ \alpha_3(P(\boldsymbol{x} - \boldsymbol{\nu}_3, t)) \cdot P(\boldsymbol{x} - \boldsymbol{\nu}_3, t) \\ &+ \alpha_4(P(\boldsymbol{x} - \boldsymbol{\nu}_4, t)) \cdot P(\boldsymbol{x} - \boldsymbol{\nu}_4, t) \\ &- k'_m \cdot F(t) \cdot P(\boldsymbol{x}, t) + k'_m \cdot F(t) \cdot P(\boldsymbol{x} - \boldsymbol{\nu}_1, t),\end{aligned}$$

which is in the form of a CME with time-varying rates (4.4). As noted in [61], this simplification is the result of special properties in the proposed abridged model. The propensity of the delayed reaction is constant, as no reactions change the number of DNA or the reaction's kinetic function. If this is not the case, then simplifying assumptions must be made for the joint probability distributions.

CHAPTER 5

SOLVING CHEMICAL MASTER EQUATION WITH TIME-VARYING RATES BY MAGNUS EXPANSION

This chapter is based on work accepted in Journal of Coupled Systems and Multiscale Dynamics [62] and work accepted in the AMMCS 2017 Proceedings [63].

5.1 Introduction

As we have discussed previously, it is difficult to solve the CME directly because it can involve a very large, or even infinite, number of ordinary differential equations and therefore has traditionally been solved indirectly by using Monte Carlo methods, such as Gillespie's Stochastic Simulation Algorithm (SSA) [64, 6] or the First Reaction Method (FRM) [65]. These algorithms simulate reactive events in the chemical processes, and averaging their results from a large number of trajectories can offer insights into the model as well as statistics of interest. Approximate representations using stochastic differential equations (SDEs) yield other approaches [66, 67].

The FSP algorithms that we have encountered so far, however, can only be applied when reaction rates are constant. In biological problems where reaction rates can change over time, for instance due to cell volume increase or change in temperature or other causes as detailed in Chapter 4, the CME may be approached with ODE solvers, such as Adams, Runge-Kutta, or backward-differentiation formula.

The Magnus expansion [68] offers an alternative approach to dealing with the CME with time-dependent rates. It expresses the solution to the ODE as the matrix exponential of an infinite series involving multiple integrals and nested commutators. Although originally a theoretical technique, it has recently been developed as a practical ODE solver [69, 70, 71, 72, 60]. The Magnus series is truncated, the terms are rearranged to

optimize the execution time, and the integrals are approximated by a quadrature formula.

In this chapter, we embed the SSA in the Magnus method to reduce the state space, which allows for lower run time while retaining the accuracy. We also revisit two error estimation techniques in the literature, as well as implement two other error control procedures, and employ these to make the Magnus method adaptive both in terms of the time step and the state space. The resulting algorithms are then tested against Adams, Runge-Kutta and backward-differentiation formula for three biological problems in which the CME with time-dependent rates emerges.

The chapter is organized as follows: Section 5.2 revisits two Monte Carlo methods commonly used to solve the CME. Some traditional ODE solvers are described in Section 5.3, and the Magnus expansion as well as the Magnus-SSA algorithm are presented in Section 5.4. Section 5.5 covers several error estimation methods, together with the resulting adaptive time-stepping Magnus algorithms. Numerical tests are defined and results are given in Section 5.6, and concluding remarks follow in Section 5.7.

For convenience, throughout this chapter, we will denote the ODE problem as

$$\dot{\boldsymbol{p}}(t) = f(t, \boldsymbol{p}(t)) \equiv \boldsymbol{A}(t) \cdot \boldsymbol{p}(t),$$

and it is worth repeating that the ODE form of the CME is:

$$\begin{cases} \dot{\boldsymbol{p}}(t) = \boldsymbol{A}(t) \cdot \boldsymbol{p}(t), \\ \boldsymbol{p}(0) = \boldsymbol{p}_0, \end{cases} \tag{5.1}$$

5.2 Monte Carlo methods

To solve the CME indirectly, Monte Carlo methods simulate trajectories from the initial state $\boldsymbol{x}(0)$ at $t = 0$ to a prescribed final time t_f. Statistics of interest are then drawn from the final states of the trajectories, such as means of the species counts, variance, or marginal distributions. Many Monte Carlo methods are based on the works of

Gillespie [64, 6]. We describe here two methods, first reaction method (FRM) and stochastic simulation algorithm (SSA).

5.2.1 First reaction method

At every time step in each trajectory, the first reaction method (FRM) [65] seeks the reaction that occurs next, and the time it takes for it to occur. Below is a pseudocode for the overall procedure:

- Step 1: Start from initial time $t = 0$ and initial state $\boldsymbol{x} = \boldsymbol{x}(0)$
- Step 2: Generate $\xi_1, \ldots, \xi_M$, uniformly distributed in $(0, 1)$
- Step 3: For every reaction k, find the minimal positive number τ_k so that

$$\int_t^{t+\tau_k} \alpha_k(\boldsymbol{x}(t), u)du = \ln\left(\frac{1}{\xi_k}\right)$$

- Step 4: τ is the minimum of $\tau_1, \ldots, \tau_M$, and j is the index of the reaction with $\tau = \tau_j$.
- Step 5: Reaction j is the first to occur next, at $t + \tau$. Update the state and time accordingly:

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\nu}_j$$
$$t \leftarrow t + \tau$$

- Step 6: Return to step 2 until reaching final time t_f.

FRM and other Monte Carlo methods approximate the probability distribution of the system at the final time t_f by simulating a large number of trajectories, and computing the frequency of each state $\boldsymbol{x}$ in the state space as

$$P(\boldsymbol{x}, t_f) \approx \text{frequency}(\boldsymbol{x}) = \frac{n_{\boldsymbol{x}}}{n_{\text{total}}}, \tag{5.2}$$

where $n_{\boldsymbol{x}}$ is the number of trajectories that end up in state $\boldsymbol{x}$, and n_{total} is the total number of trajectories.

FRM is an exact method, because the trajectories are generated according to the correct probability distributions. For each problem in our numerical tests, we use the FRM frequencies to compare the results of the ODE solvers.

5.2.2 Stochastic simulation algorithm

If the reaction rates are time-independent, the first reaction method and the stochastic simulation algorithm (SSA) [64, 6] are equivalent and the pseudocode simplifies to:

- Step 1: Start from initial time $t = 0$ and initial state $\boldsymbol{x} = \boldsymbol{x}(0)$
- Step 2: Find the propensity sum $\alpha_{sum} = \sum_{k=1}^{M} \alpha_k(\boldsymbol{x}(t), t)$
- Step 3: Generate ξ_1 and ξ_2, uniformly distributed in $(0, 1)$
- Step 4: j is the smallest integer so that

$$\sum_{k=1}^{j} \alpha_k(\boldsymbol{x}, t) > \xi_1 \alpha_{sum}$$

- Step 5: Compute $\tau = \ln\left(\frac{1}{\xi_2}\right) \alpha_{sum}$
- Step 6: Reaction j is the first to occur next, at $t + \tau$. Update the state and time accordingly:

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\nu}_j$$

$$t \leftarrow t + \tau$$

- Step 7: Return to step 2 until reaching final time t_f.

Note that the SSA is inexact for our purpose, because it does not account for the time dependencies of the reaction rates. However, SSA is much faster than FRM, because the latter contains M optimization problems and possibly many integration problems at every time step.

5.3 ODE solvers

We will now describe several traditional ODE solvers for the purpose of solving the CME directly. A detailed introduction to different classes of ODE solvers can be found in [73, 74, 75]. Here we summarize the ones used in our numerical tests.

5.3.1 Adams

Adams methods form a family of linear multi-step methods [76], among which are explicit Adams-Bashforth and implicit Adams-Moulton. Adams-Bashforth proceeds with the explicit formulae of order r:

$$\begin{aligned} t_{k+1} &= t_k + h_k, \\ \boldsymbol{p}_{k+1} &= \boldsymbol{p}_k + h_k \left(\beta_{r-1}^{AB} \boldsymbol{f}_k + \cdots + \beta_0^{AB} \boldsymbol{f}_{k-r+1} \right), \\ \boldsymbol{f}_{k+1} &= f(t_{k+1}, \boldsymbol{p}_{k+1}), \end{aligned}$$

where $\left\{ \beta_i^{AB} \right\}_{i=0}^{r-1}$ are given analytically based on a Lagrange interpolation polynomial.

Adams-Mouton, on the other hand, is implemented as

$$\begin{aligned} t_{k+1} &= t_k + h_k, \\ \boldsymbol{p}_{k+1} &= \boldsymbol{p}_k + h_k \left(\beta_{r}^{AM} \boldsymbol{f}_{k+1} + \cdots + \beta_0^{AM} \boldsymbol{f}_{k-r+1} \right), \\ \boldsymbol{f}_{k+1} &= f(t_{k+1}, \boldsymbol{p}_{k+1}), \end{aligned}$$

where $\left\{ \beta_i^{AM} \right\}_{i=0}^{r}$ are given analytically.

We use the **Adams-PECE** scheme by Shampine and Gordon [77], which

implements the implicit Adams-Moulton. The unknown $\boldsymbol{p}_{k+1} \approx \boldsymbol{p}(t_{k+1})$ is involved in both sides of the formula, leading to a nonlinear problem that is approximately solved with a fixed-point scheme starting from the solution of the explicit Adams-Bashforth.

5.3.2 Runge-Kutta

Runge-Kutta methods form a class of multistage, one-step iteration ODE solvers. The explicit Runge-Kutta of order r proceeds with the scheme

$$\begin{aligned} t_{k+1} &= t_k + h_k, \\ \boldsymbol{y}_i &= \boldsymbol{p}_k + h_k \sum_{j=1}^{i-1} m_{ij}^{RK} f(t_k + h_k c_j^{RK}, \boldsymbol{y}_j); i = 1, \ldots, r, \\ \boldsymbol{p}_{k+1} &= \boldsymbol{p}_k + h_k \sum_{j=1}^{r} b_j^{RK} f\left(t_k + h_k c_j^{RK}, \boldsymbol{y}_j\right), \end{aligned}$$

in which the coefficients $\left\{m_{ij}^{RK}\right\}_{i,j=1}^{s}$, $\left\{b_i^{RK}\right\}_{i=1}^{s}$ and $\left\{c_i^{RK}\right\}_{i=1}^{s}$ are defined by their Butcher-tableau.

In this comparison, we use the solver RK78 in the RKSUITE by Brankin et al. [78], which is a reputed Runge-Kutta method that controls the error and stepsize by using embedded Runge-Kutta formulae with orders 7 and 8. The solutions are denoted as **Runge-Kutta** in the numerical tests.

5.3.3 Backward-differentiation formula

Backward-differentiation formula (BDF) methods are linear multi-step and follow the formula of order r:

$$\begin{aligned} t_{k+1} &= t_k + h_k, \\ \boldsymbol{p}_{k+1} &= h_k \beta_r^{BDF} f(t_{k+1}, \boldsymbol{p}_{k+1}) + \\ &\quad \alpha_{r-1}^{BDF} \boldsymbol{f}_k + \cdots + \alpha_0^{BDF} \boldsymbol{f}_{k-r+1}, \\ \boldsymbol{f}_{k+1} &= f(t_{k+1}, \boldsymbol{p}_{k+1}), \end{aligned}$$

where the coefficients $\left\{\alpha_i^{BDF}\right\}_{i=0}^{r-1}$ and β_r^{BDF} are given analytically. The formula forms a nonlinear problem, because $\boldsymbol{p}_{k+1}$ appears on both sides.

We use the VODPK/BDF implementation [79] which has different options in solving the nonlinear problem. Here we confine to:

- **BDF-GM-LU0**: Newton root finding scheme; each linear system in the implicit scheme is solved iteratively by SPIGMR (Scaled Preconditioned Incomplete GMRES), preconditioned by the incomplete LU0 decomposition, which discards elements not in the sparsity pattern of $\boldsymbol{A}$,
- **BDF-LU0**: the linear system in the impiclit scheme is solved directly by incomplete LU0 decomposition.

5.4 Magnus-based methods

5.4.1 Magnus expansion

The Magnus expansion [68, 69, 70] expresses the theoretical solution of (5.1) as

$$\boldsymbol{p}(t) = \exp(\boldsymbol{\Omega}_{(0,t)}) \cdot \boldsymbol{p}_0, \tag{5.3}$$

where the matrix exponential is defined as

$$\exp(\boldsymbol{\Omega}) = \sum_{l=0}^{\infty} \frac{\boldsymbol{\Omega}^l}{l!},$$

and $\boldsymbol{\Omega}_{(0,t)}$ can be written as an infinite series whose terms involve multiple integrals and nested commutators:

$$\begin{aligned}
\boldsymbol{\Omega}_1 &= \int_0^t \boldsymbol{A}(\tau) d\tau, \\
\boldsymbol{S}_n^{(1)} &= [\boldsymbol{\Omega}_{n-1}, \boldsymbol{A}], \\
\boldsymbol{S}_n^{(j)} &= \sum_{m=1}^{n-j} [\boldsymbol{\Omega}_m, \boldsymbol{S}_m^{j-1}]; 2 \leq j \leq n-1,
\end{aligned}$$

$$
\begin{aligned}
\boldsymbol{S}_n^{(n-1)} &= \mathrm{ad}_{\Omega_1}^{n-1}(\boldsymbol{A}), \\
\boldsymbol{\Omega}_n &= \sum_{j=1}^{n-1} \frac{B_j}{j!} \int_0^t \boldsymbol{S}_n^{(j)}(\tau) d\tau; n \geq 2, \\
\boldsymbol{\Omega}_{(0,t)} &= \sum_{n=1}^{\infty} \boldsymbol{\Omega}_n,
\end{aligned} \tag{5.4}
$$

where B_j are Bernoulli numbers and the adjoint representation is defined as

$$
\mathrm{ad}_{\boldsymbol{A}}^j(\boldsymbol{B}) = [\boldsymbol{A}, \mathrm{ad}_{\boldsymbol{A}}^{j-1}(\boldsymbol{B})], \ \mathrm{ad}_{\boldsymbol{A}}^0(\boldsymbol{B}) = \boldsymbol{B}.
$$

The Magnus integration method then follows:

$$
\begin{aligned}
t_{k+1} &= t_k + h_k, \\
\boldsymbol{p}_{k+1} &= \exp\left(\boldsymbol{\sigma}_{(t_k,h_k)}^{[q]}\right) \cdot \boldsymbol{p}_k,
\end{aligned} \tag{5.5}
$$

where $\boldsymbol{\sigma}_{(t_k,h_k)}^{[q]}$ is an approximation of $\boldsymbol{\Omega}_{(t_k,h_k)}$ to the qth order. To derive $\boldsymbol{\sigma}_{(t_k,h_k)}^{[q]}$, the Magnus expansion (5.4) is truncated to the first q terms, and the integrals are approximated by a quadrature rule.

The Magnus expansion has been employed extensively in physics where it is sometimes referred to as time-dependent exponential perturbation theory [70]. It has the appeal that its approximation preserves important qualitative properties of the exact solution [72, 80]. It has also been used in the field of geometric numerical integration to reproduce important geometric structures in the solutions, a goal not straightforwardly possible with general-purpose integration methods.

We implement the 4th-order Magnus integration method using the Gauss-Legendre

quadrature rule [71, 69, 81]:

$$\begin{aligned}
\boldsymbol{A}_1 &= \boldsymbol{A}\left(t_k + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right) h_k\right), \\
\boldsymbol{A}_2 &= \boldsymbol{A}\left(t_k + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right) h_k\right), \\
\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)} &= \frac{h_k}{2}\left(\boldsymbol{A}_1 + \boldsymbol{A}_2\right) + \frac{h_k^2\sqrt{3}}{12}\left[\boldsymbol{A}_2, \boldsymbol{A}_1\right].
\end{aligned}$$

5.4.2 Krylov subspace technique

The term $\exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}\right) \cdot \boldsymbol{p}_k$ in (5.5) is computed by Expokit, which implements a Krylov-based algorithm that seeks to approximate $\exp\left(\boldsymbol{\sigma}\right) \cdot \boldsymbol{p}$, the action of the matrix exponential on a vector, as a projection in the Krylov subspace of order m

$$\mathcal{K}_m(\boldsymbol{\sigma}, \boldsymbol{p}) = \text{span}\left\{\boldsymbol{p}, \boldsymbol{\sigma} \cdot \boldsymbol{p}, \ldots, \boldsymbol{\sigma}^{m-1} \cdot \boldsymbol{p}\right\}.$$

The Arnoldi process is employed to compute an orthonormal basis $\boldsymbol{V}_m$ of this subspace, and an associated Hessenberg matrix $\boldsymbol{H}_m$. The Krylov approximation is then

$$\exp\left(\boldsymbol{\sigma}\right) \cdot \boldsymbol{p} \approx \beta \boldsymbol{V}_m \exp(\boldsymbol{H}_m)\boldsymbol{e}_1,$$

where $\beta = \|\boldsymbol{p}\|_2$ and $\boldsymbol{e}_1 = (1, 0, \ldots, 0)^T$. The Padé approximation [82], together with scaling and squaring, is employed to compute $\exp(\boldsymbol{H}_m)$. Other variants with an incomplete orthogonalization in the Arnoldi process could also be attempted [83], but this was not our focus here.

Krylov-based methods have proved efficient when the matrix is sparse, as is the case in many biological problems. They also have the appeal of being matrix-free by only requiring the matrix-vector product $\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)} \cdot \boldsymbol{v}$ to compute $\exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}\right) \cdot \boldsymbol{p}_k$. Our numerical tests use the vanilla Expokit with Krylov order $m = 30$, producing approximations well within the tolerance $tol = 10^{-5}$ even for problems with large sizes.

5.4.3 Magnus with an adaptive SSA-based state space

During the integration time of any ODE solver for (5.1), most of the values in $\boldsymbol{p}(t)$ will be extremely small and therefore computing the full distribution can be expensive without gaining much accuracy. For CME problems with time-independent rates, the FSP-SSA method [1] confines the state space $\boldsymbol{X}$ at each step to only the most probable states at that step. To apply this strategy in our context, we proceed as follows: knowing the current $\boldsymbol{p}_k \approx \boldsymbol{p}(t_k)$ and having chosen a step-size h_k, we wish to advance to the next approximation $\boldsymbol{p}_{k+1} \approx \boldsymbol{p}(t_k + h_k)$ using (5.5) with $\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}$ restricted to only the subset of states that the system is likely to occupy during the time interval $[t_k, t_k + h_k]$. This is implemented by first dropping current states with low probabilities in $\boldsymbol{p}_k$, then sampling SSA trajectories from the remaining states and updating it to include all states that the SSA paths travel through. The 'roughness' in the paths is then smoothed out by the r-step reachability [10], which seeks all states that can be connected to the state space by r reactions or less, and expands the state space to include those. Below is the corresponding pseudocode.

- Step 1: $\boldsymbol{X}$ is reduced to states with probability $> 10^{-16}$
- Step 2: $\boldsymbol{X}$ is expanded by SSA runs over $[t_k, t_k + h_k]$ and r-step reachability with $r = 5$
- Step 3: $\boldsymbol{A}(t)$ is updated to only states in $\boldsymbol{X}$
- Step 4: Compute $\boldsymbol{p}_{k+1} = \exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}\right) \cdot \boldsymbol{p}_k$

The implementation uses a hash table to keep track of the state space and also note that $\boldsymbol{A}(t)$ is in functional form for the matrix-free Krylov stage, with the action of the matrix exponential in the last step done by using Expokit with the matrix-vector operator

$\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)} \cdot \boldsymbol{v}$ defined through [81]:

$$
\begin{aligned}
\boldsymbol{A}_1 &= \boldsymbol{A}\left(t_k + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right) h_k\right), \\
\boldsymbol{A}_2 &= \boldsymbol{A}\left(t_k + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right) h_k\right), \\
\boldsymbol{w}_1 &= \boldsymbol{A}_1 \boldsymbol{v}, \\
\boldsymbol{w}_2 &= \boldsymbol{A}_2 \boldsymbol{v}, \\
\boldsymbol{w}_3 &= \boldsymbol{A}_2 \boldsymbol{w}_1, \\
\boldsymbol{w}_4 &= \boldsymbol{A}_1 \boldsymbol{w}_2, \\
\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)} \cdot \boldsymbol{v} &= \frac{h_k}{2}\left(\boldsymbol{w}_1 + \boldsymbol{w}_2\right) + \frac{h_k^2\sqrt{3}}{12}\left(\boldsymbol{w}_3 - \boldsymbol{w}_4\right).
\end{aligned}
$$

It is important to mention again that SSA is considered inexact for the CME with time-dependent rates, because reaction rates are kept constant during each time step. However, the SSA is used only to expand the state space. The probability distribution is computed using the Magnus method instead, therefore the results are not compromised. In the cases where the reaction rates change dramatically, the state space can be expanded using the FRM. This will be more time-consuming, but the state spaces will not be distorted.

The step-size h_k can be kept constant throughout the integration [63]. In this case, the choice of the step-size is very important for the performance of the Magnus-SSA algorithm, as the resulting probability distributions can be distorted if the step-size is too large, and the algorithm would be too time-consuming if the step-size is too small. We therefore implement an adaptive **MAGNUS-SSA** method, which requires a scheme for computing the step-size h_k. This will be completed in the next section.

5.5 Adaptive time-stepping schemes

As can be seen from the previous section, the error in **MAGNUS-SSA** comes from a combination of four different error sources:

- The FSP error: from truncating the state space
- The Magnus truncation error: from truncating the infinite Magnus series
- The quadrature error: from approximating the integrals involved with quadrature rules
- The Krylov error: from approximating $\boldsymbol{p}_{k+1} = \exp(\boldsymbol{\sigma}) \cdot \boldsymbol{p}_k$ using Krylov subspace techniques.

The FSP error exists for all ODE solvers and **MAGNUS-SSA**. However, in our numerical tests, for the ODE solvers, a sufficiently big state space was truncated at the beginning of each algorithm according to a large number of SSA trajectories, and is fixed during the entire integration. The FSP error for these schemes is therefore minimal. In **MAGNUS-SSA**, the state space is changed at each time step, but the large number of SSA trajectories and r-step reachability with large r required to build the state space assure that the FSP error is also insignificant. Therefore we assume that the FSP error is negligible for all algorithms.

The Krylov error, on the other hand, is automatically managed by Expokit [37], which is the most extensive software for computing the matrix exponential and has been reported to be well suited for large sparse matrices [82]. Expokit allows an error tolerance, so we can also leave out the Krylov error for simplicity.

The dominant error of the Magnus-based methods, therefore, comes from truncating and interpolating the Magnus series. We seek to approximate this error and use it in an adaptive time-stepping Magnus-based scheme.

5.5.1 Adaptive time-stepping scheme for MAGNUS-SSA

At any time interval $[t_k, t_k + h_k]$, the local error is defined to be

$$\text{error}(t_{k+1}) = \|\boldsymbol{p}_{k+1} - \bar{\boldsymbol{p}}_{k+1}\|_1,$$

where

$$\boldsymbol{p}_{k+1} = \exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}\right) \cdot \boldsymbol{p}_k$$

is the **MAGNUS-SSA** approximation to the exact solution

$$\bar{\boldsymbol{p}}_{k+1} = \exp(\boldsymbol{\Omega}_{(t_k,h_k)}) \cdot \boldsymbol{p}_k.$$

The following pseudocode is a template for the overall step-by-step integration process using a traditional step-size control.

Algorithm MAGNUS-SSA

1: Initialize state space and $\boldsymbol{p}_0$ and time $t_0 = 0$
2: Initialize step-size $h_0 = 0.5$ and step $k = 0$
3: **while** $t_k < t_f$ **do**
4: Update state space and get $\boldsymbol{p}_{k+1} = \exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}\right)\boldsymbol{p}_k$
as discussed in the previous section
5: Compute the error estimate $\text{error}(t_{k+1})$
6: **if** $\text{error}(t_{k+1}) > \epsilon$ **then**
7: $h_k \leftarrow 0.5 \cdot h_k$
8: **go to** Step 3
9: **end if**
10: $t_{k+1} \leftarrow t_k + h_k$
11: $h_{k+1} \leftarrow \delta \left(\frac{\epsilon}{\text{error}}\right)^{1/p} h_k$
12: $k \leftarrow k + 1$
13: **end while**

The safety factor δ is 0.5 in our code. The error tolerance is $\epsilon = 10^{-5}$, and Expokit is implemented with the same tolerance. The parameter p in Step 11 is set as the order of the error estimator, to be discussed next. The discussed variants use the same Magnus approximation in Step 3 and only differ in how they perform the error estimation in Step 5

of the template. We will see later that Step 3 is moved to be after the if-block in the case of **MAGNUS-SSA-4** because it has an *a priori* error control.

5.5.2 Error approximation

A traditional error estimating technique is to compute the results of one ODE solver with two different orders, and compute the difference. MacNamara and Burrage [81] developed a local error estimate based on this embedding scheme:

$$\begin{aligned}\text{error}(t_{k+1}) &\approx \left\| \exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}\right)\cdot \boldsymbol{p}_k - \exp\left(\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)}\right)\cdot \boldsymbol{p}_k \right\|_1 \\ &= \left\| \boldsymbol{p}_{k+1} - \exp\left(\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)}\right)\cdot \boldsymbol{p}_k \right\|_1,\end{aligned}$$

where $\exp\left(\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)}\right)\cdot \boldsymbol{p}_k$ is the 2nd-order Magnus with Gauss-Legendre quadrature rule, computed with Expokit where the matrix-vector product operator $\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)}\cdot \boldsymbol{v}$ defined as

$$\begin{aligned}\boldsymbol{A}_1 &= \boldsymbol{A}\left(t_k + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right)h_k\right),\\ \boldsymbol{A}_2 &= \boldsymbol{A}\left(t_k + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)h_k\right),\\ \boldsymbol{w}_1 &= \boldsymbol{A}_1\boldsymbol{v},\\ \boldsymbol{w}_2 &= \boldsymbol{A}_2\boldsymbol{v},\\ \boldsymbol{\sigma}^{[2]}_{(t_k,h_k)}\cdot \boldsymbol{v} &= \frac{h_k}{2}\left(\boldsymbol{w}_1 + \boldsymbol{w}_2\right).\end{aligned}$$

Since this error estimate is based on the 2nd-order Magnus approximation, it is of order $p = 2$. The algorithm will proceed with the 4th-order Magnus approximation instead. This Magnus implementation is denoted **MAGNUS-SSA-1** in our numerical tests.

At each time step, **MAGNUS-SSA-1** requires two Expokit runs, one for the 2nd-order solution and one for the 4th-order solution, where only the latter is required to proceed. This can be time-consuming, therefore we propose a new cheaper local error estimate that removes the need to explicitly compute the 2nd-order approximation. The

starting point is the observation that the inverse of a matrix exponential is

$$[\exp(\boldsymbol{\sigma})]^{-1} = \exp(-\boldsymbol{\sigma}),$$

which allows us to rewrite the terms in error(t_{k+1}) as

$$\left[\exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}\right) - \exp\left(\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)}\right)\right] \cdot \boldsymbol{p}_k = \left[\boldsymbol{I} - \exp\left(\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)}\right)\exp\left(-\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}\right)\right] \cdot \boldsymbol{p}_{k+1} \quad (5.6)$$

From this, we use the fact that the product of two matrix exponentials can be approximated using the Baker-Campbell-Hausdorff formula, and the important fact that the matrix-vector product for the 2nd-order Magnus algorithm is embedded in that for the 4th-order Magnus algorithm. This reduces (5.6) into

$$\begin{aligned}
&\left[\boldsymbol{I} - \exp\left(\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)} - \boldsymbol{\sigma}^{[4]}_{(t_k,h_k)} - \frac{1}{2}\left[\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)}, \boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}\right] + \mathcal{O}(h_k^4)\right)\right] \cdot \boldsymbol{p}_{k+1} \\
&= \left[-\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)} + \boldsymbol{\sigma}^{[4]}_{(t_k,h_k)} + \frac{1}{2}\left[\boldsymbol{\sigma}^{[2]}_{(t_k,h_k)}, \boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}\right] + \mathcal{O}(h_k^4)\right] \cdot \boldsymbol{p}_{k+1} \\
&= \left[\frac{h_k^2\sqrt{3}}{12}[\boldsymbol{A}_2, \boldsymbol{A}_1] + \frac{h_k^3\sqrt{3}}{48}[\boldsymbol{A}_1 + \boldsymbol{A}_2, [\boldsymbol{A}_2, \boldsymbol{A}_1]] + \mathcal{O}(h_k^4)\right] \cdot \boldsymbol{p}_{k+1},
\end{aligned}$$

where $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are defined as in the **MAGNUS-SSA** algorithm.

We can derive from this a cheap approximation of the local error estimate in

MAGNUS-SSA-1:

$$\boldsymbol{u}_1 = \boldsymbol{A}_1 \cdot \boldsymbol{p}_{k+1}; \quad \boldsymbol{u}_2 = \boldsymbol{A}_2 \cdot \boldsymbol{p}_{k+1}; \quad \boldsymbol{u}_3 = \boldsymbol{A}_2 \cdot \boldsymbol{u}_1;$$

$$\boldsymbol{u}_4 = \boldsymbol{A}_1 \cdot \boldsymbol{u}_2; \quad \boldsymbol{u}_5 = \boldsymbol{A}_1 \cdot \boldsymbol{u}_1; \quad \boldsymbol{u}_6 = \boldsymbol{A}_2 \cdot \boldsymbol{u}_2;$$

$$\boldsymbol{u}_7 = \boldsymbol{A}_1 \cdot \boldsymbol{u}_3; \quad \boldsymbol{u}_8 = \boldsymbol{A}_2 \cdot \boldsymbol{u}_4; \quad \boldsymbol{u}_9 = \boldsymbol{A}_2 \cdot \boldsymbol{u}_3;$$

$$\boldsymbol{u}_{10} = \boldsymbol{A}_1 \cdot \boldsymbol{u}_4; \quad \boldsymbol{u}_{11} = \boldsymbol{A}_1 \cdot \boldsymbol{u}_6; \quad \boldsymbol{u}_{12} = \boldsymbol{A}_2 \cdot \boldsymbol{u}_5;$$

$$\text{error}(t_{k+1}) = \left\| \frac{h_k^2\sqrt{3}}{12}(\boldsymbol{u}_3 - \boldsymbol{u}_4) + \frac{h_k^3\sqrt{3}}{48}(2\boldsymbol{u}_7 - 2\boldsymbol{u}_8 + \boldsymbol{u}_9 - \boldsymbol{u}_{10} + \boldsymbol{u}_{11} - \boldsymbol{u}_{12}) \right\|_1 .$$

The Magnus implementation using this error estimate is denoted **MAGNUS-SSA-2** in the numerical tests. Because it is based on **MAGNUS-SSA-1**, the error is also of order $p = 2$.

Another well-known error approximating approach is computing the difference between the result of the ODE solver with the result of the same solver with halved time-step. Using this technique, the local error is then:

$$\begin{aligned} \text{error}(t_{k+1}) &\approx \left\| \exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)}\right) \cdot \boldsymbol{p}_k - \exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k+h_k/2,h_k/2)}\right) \cdot \exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k/2)}\right) \cdot \boldsymbol{p}_k \right\|_1 \\ &= \left\| \boldsymbol{p}_{k+1} - \exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k+h_k/2,h_k/2)}\right) \cdot \exp\left(\boldsymbol{\sigma}^{[4]}_{(t_k,h_k/2)}\right) \cdot \boldsymbol{p}_k \right\|_1 . \end{aligned}$$

The Magnus implementation using this error estimate is called **MAGNUS-SSA-3** in the numerical tests. A disadvantage of this approach is that Expokit has to be run three times for every time step, one for the normal 4th-order Magnus approximation and two for the 'corrector'. On the other hand, the error estimate is of order $p = 4$, therefore the time steps will be less conservative than **MAGNUS-SSA-1** and **MAGNUS-SSA-2**.

The final approach of approximating the local error in consideration is based on the leading term of the error in truncating the Magnus expansion. Iserles, Marthinsen and

Nørsett [71] derived

$$\boldsymbol{\sigma}^{[4]}_{(t_k,h_k)} = \boldsymbol{\Omega}_{(t_k,h_k)} + \frac{h_k^4}{720}\left[\boldsymbol{A}(t_k+h_k),[\boldsymbol{A}(t_k+h_k),[\boldsymbol{A}(t_k),\boldsymbol{A}(t_k+h_k)]]\right] + \mathcal{O}(h_k^5).$$

They used the Baker-Campbell-Hausdorff formula to obtain a local error estimate of order $p = 4$ from this equation:

$$\text{error}(t_{k+1}) = \left\| \frac{h_k^4}{720}\left[\boldsymbol{A}(t_k+h_k),[\boldsymbol{A}(t_k+h_k),[\boldsymbol{A}(t_k),\boldsymbol{A}(t_k+h_k)]]\right]\cdot \boldsymbol{p}_k \right\|_1 .$$

Expanding the nested commutators in the equation and collecting like-terms, we get a commutator-free form of this error estimate, used in **MAGNUS-SSA-4**:

$$\begin{aligned}
\boldsymbol{A}_0 &= \boldsymbol{A}\left(t_k\right), \\
\boldsymbol{A}_3 &= \boldsymbol{A}\left(t_k+h_k\right), \\
\boldsymbol{u}_1 &= \boldsymbol{A}_3\cdot\boldsymbol{A}_3\cdot\boldsymbol{A}_0\cdot\boldsymbol{A}_3\cdot\boldsymbol{p}_k, \\
\boldsymbol{u}_2 &= \boldsymbol{A}_3\cdot\boldsymbol{A}_0\cdot\boldsymbol{A}_3\cdot\boldsymbol{A}_3\cdot\boldsymbol{p}_k, \\
\boldsymbol{u}_3 &= \boldsymbol{A}_3\cdot\boldsymbol{A}_3\cdot\boldsymbol{A}_3\cdot\boldsymbol{A}_0\cdot\boldsymbol{p}_k, \\
\boldsymbol{u}_4 &= \boldsymbol{A}_0\cdot\boldsymbol{A}_3\cdot\boldsymbol{A}_3\cdot\boldsymbol{A}_3\cdot\boldsymbol{p}_k, \\
\text{error}(t_{k+1}) &= \left\| \frac{h_k^4}{720}\left(3\boldsymbol{u}_1 - 3\boldsymbol{u}_2 - \boldsymbol{u}_3 + \boldsymbol{u}_4\right)\right\|_1 .
\end{aligned}$$

A great advantage of **MAGNUS-SSA-4** over the other Magnus variants in our comparison is that the error estimate is *a priori*. Its implementation moves Step 3 to be after the if-block in the template given earlier so that for each time step, it calculates the error before committing to proceed, and therefore does not run Expokit unnecessarily. There is also only one Expokit run per time step, keeping the execution time minimal.

As implemented here, this error estimate has the shortcoming that it takes into account only the Magnus truncation error and not the quadrature error, unlike the other

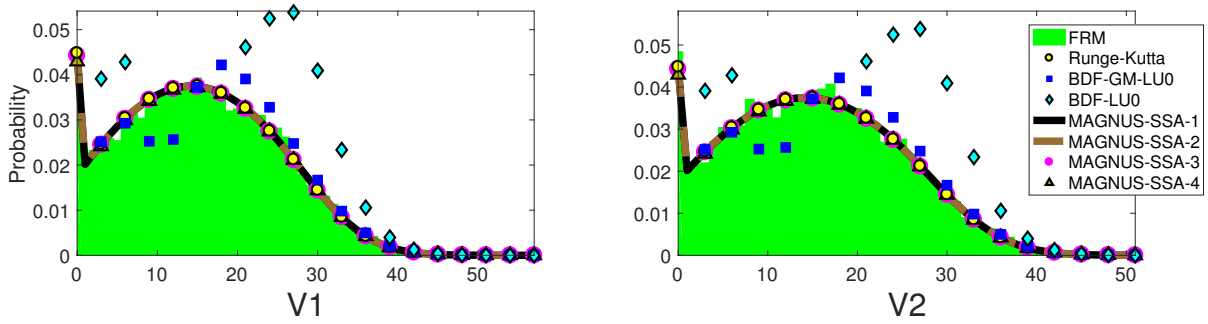


Figure 5.1: Probability distributions from the two competing T cell clonotypes (Test 1)

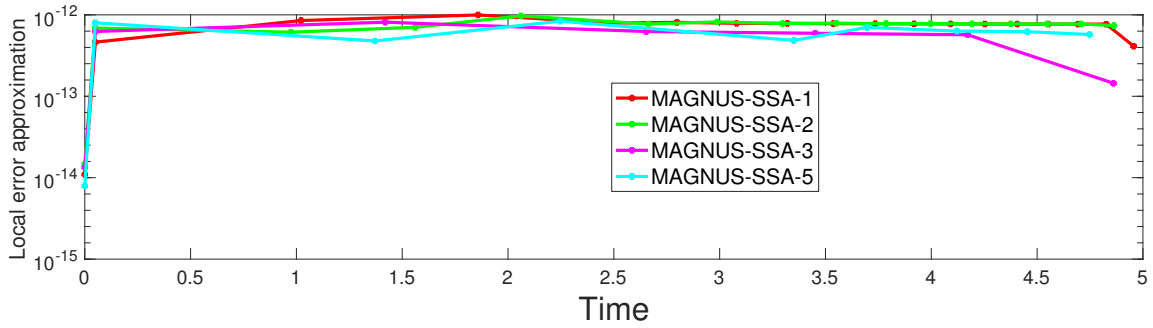


Figure 5.2: Error estimates in the **MAGNUS-SSA** variants from the two competing T cell clonotypes (Test 1)

Magnus variants. The quadrature error is notoriously difficult to estimate [84, 85], and it depends on the behavior of $\boldsymbol{A}(t)$.

5.6 Numerical tests

All numerical tests were done using resources of the Alabama Supercomputer, which houses two supercomputers called SGI UV and DMC. The user can request a job to be executed on either of them, or can simply let the operating system select the more suitable system depending on the workload and availability. All codes were written in FORTRAN 77 and were run on the large queue of the SGI UV with 1 processor core (Xeon E5-4640 CPU operating at 2.4 GHz), 360hr time limit and 120GB memory limit.

The models in our numerical tests arise from different fields of biology. In each numerical test, the distributions from solving (5.1) by the ODE solvers are compared with the frequencies from 10,000 FRM trajectories, computed by (5.2). The fixed FSP state

space for the ODE solvers is found by finding the maximum and minimum of each species count during the 10,000 FRM trajectories, except for **MAGNUS-SSA**, which does not require *a priori* fixed FSP bounds and changes the state space adaptively instead. These FSP bounds are reported for each numerical test.

The error of each ODE solver is defined to be the maximum of 1-norm differences between the marginal distributions from that ODE solver and the FRM.

5.6.1 Test 1 - the model of two competing T cell clonotypes

The first problem in our comparison models the competition between T cell clonotypes [55, 81]. We consider two species:

$$\mathrm{V}_1 : \text{T cell clonotype 1},$$
$$\mathrm{V}_2 : \text{T cell clonotype 2},$$

which can interact through a multivariate birth-death process:

$$\varnothing \xrightarrow{c_1} \mathrm{V}_1,$$
$$\mathrm{V}_1 \xrightarrow{c_2} \varnothing,$$
$$\varnothing \xrightarrow{c_3} \mathrm{V}_2,$$
$$\mathrm{V}_2 \xrightarrow{c_4} \varnothing.$$

The reaction rates are [81]:

$$
\begin{aligned}
c_1 &= \phi(t) \cdot \left(\frac{0.5[\mathrm{V}_1]}{[\mathrm{V}_1]+[\mathrm{V}_2]} + \frac{0.5[\mathrm{V}_1]}{[\mathrm{V}_1]+1000} \right) \ (\mathrm{year}^{-1}), \\
c_2 &= 1 \ (\mathrm{year}^{-1}), \\
c_3 &= \phi(t) \cdot \left(\frac{0.5[\mathrm{V}_2]}{[\mathrm{V}_1]+[\mathrm{V}_2]} + \frac{0.5[\mathrm{V}_2]}{[\mathrm{V}_2]+1000} \right) \ (\mathrm{year}^{-1}), \\
c_4 &= 1 \ (\mathrm{year}^{-1}),
\end{aligned}
$$

where $\phi(t) = \frac{60}{1+\left(\frac{t}{15}\right)^5}$ models the decreasing stimulation that the clonotypes receive.

We start from the initial values

$$
\begin{aligned}
\mathrm{V}_1 &= 10 \\
\mathrm{V}_2 &= 10
\end{aligned}
$$

until the end time $t_f = 5$ (years). The FRM trajectories during this time interval result in the bounds for the FSP state space:

$$
\begin{aligned}
0 \leq \ & \mathrm{V}_1 \ \leq 57, \\
0 \leq \ & \mathrm{V}_2 \ \leq 51.
\end{aligned}
$$

The FSP state space contains $n = 3016$ states and the CME matrix contains $nz = 14860$ nonzero elements. The infinity norm of the CME matrix at $t = 0$ is 513603.

The results from the ODE solvers are summarized in table 5.1. Figure 5.1 shows the probability distributions from these ODE solvers, and figure 5.2 shows the local error estimates from all **MAGNUS-SSA** algorithms. Figure 5.1 shows that the solutions from **Runge-Kutta** and the **MAGNUS-SSA** implementations agree with the FRM frequencies. **Adams-PECE** did not finish, while the solutions of both **BDF-GM-LU0** and **BDF-LU0** fail to follow the shape of the correct distributions, resulting in noticeable errors, as seen in table 5.1. Figure 5.2 shows that the error estimates from all

Table 5.1: Results from the ODE solvers for the model of two competing T cell clonotypes (Test 1). The error is computed as the maximum 1-norm difference between the marginal distributions from each ODE solver and the FRM.

ODE solver	Time cost and note	Error
Adams-PECE	Incorrect solver (flag 4)	N/A
Runge-Kutta	1s	0.0481
BDF-GM-LU0	1s	0.1624
BDF-LU0	5s	1.3152
MAGNUS-SSA-1	10s	0.0478
MAGNUS-SSA-2	8s	0.0476
MAGNUS-SSA-3	4s	0.0479
MAGNUS-SSA-4	3s	0.0491

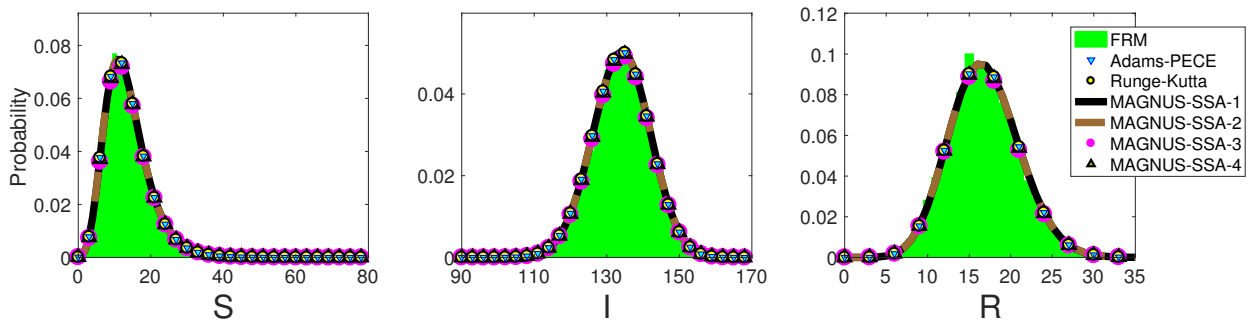


Figure 5.3: Probability distributions from the SIR model with periodic contact rate (Test 2)

MAGNUS-SSA variants are very similar.

5.6.2 Test 2 - the epidemic model with periodic contact rate

The second problem in our comparison is an epidemic model [56, 57, 58, 59], consisting of three species:

S : susceptible population,

I : infected population,

R : recovered population.

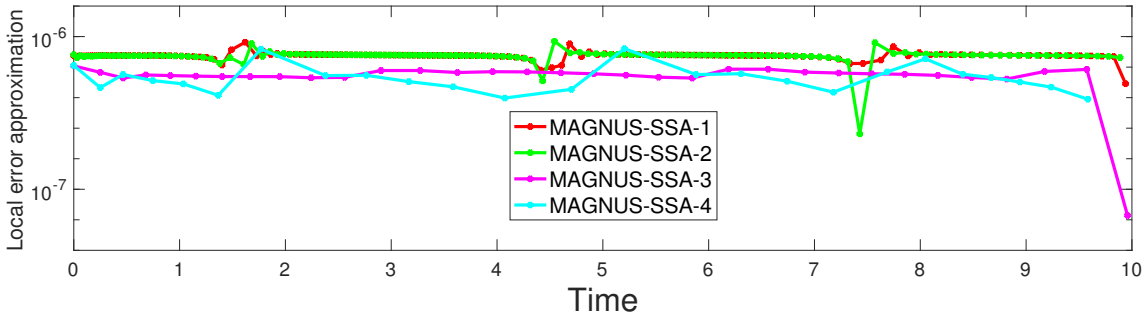


Figure 5.4: Error estimates in the **MAGNUS-SSA** variants from the SIR model with periodic contact rate (Test 2)

These populations can interact in six different reactions:

$$\mathrm{S} + \mathrm{I} \xrightarrow{c_1} 2\mathrm{I},$$

$$\mathrm{I} \xrightarrow{c_2} \mathrm{R},$$

$$\mathrm{R} \xrightarrow{c_3} \mathrm{S},$$

$$\mathrm{S} \xrightarrow{c_4} \varnothing,$$

$$\mathrm{I} \xrightarrow{c_5} \varnothing,$$

$$\mathrm{R} \xrightarrow{c_6} \varnothing,$$

with reaction rates

$$c_1 = 0.003 f(t) \ (\mathrm{day}^{-1}),$$

$$c_2 = 0.02 \ (\mathrm{day}^{-1}),$$

$$c_3 = 0.007 \ (\mathrm{day}^{-1}),$$

$$c_4 = 0.002 \ (\mathrm{day}^{-1}),$$

$$c_4 = 0.05 \ (\mathrm{day}^{-1}),$$

$$c_6 = 0.002 \ (\mathrm{day}^{-1}),$$

where $f(t) = \left(1 + 0.6 \sin\left(\frac{2\pi \cdot t}{6}\right)\right)$ describes the periodic contact rate between the infected population and the susceptible population.

The initial values for the populations are:

$$S = 200,$$
$$I = 10,$$
$$R = 0,$$

and the end time is $t_f = 10$ (years). The bounds for the FSP state space are:

$$\begin{aligned} 0 \leq \quad \mathrm{S} \quad &\leq 200, \\ 0 \leq \quad \mathrm{I} \quad &\leq 17, \\ 0 \leq \quad \mathrm{R} \quad &\leq 35. \end{aligned}$$

There are $n = 1237356$ states in the FSP state space, and $nz = 8518611$ nonzero elements in the CME matrix. The infinity norm of the CME matrix is 78286181 at $t = 0$.

Table 5.2 summarizes the results from the ODE solvers. Figures 5.3 and 5.4 show the probability distributions from the ODE solvers and the error estimates in the **MAGNUS-SSA** variants. Because the solutions from **BDF-GM-LU0** and **BDF-LU0** are very far from the correct probability distributions and even contain negative values (see table 5.2), their approximations are not included in Figure 5.3. The solutions from **Adams-PECE**, **Runge-Kutta** and all **MAGNUS-SSA** variants are shown in agreement with the FRM frequencies. Figure 5.4 shows that the error estimates are periodic, with slight disturbances when $f(t)$ reaches maximum or minimum.

Table 5.2: Results from the ODE solvers for the epidemic model with periodic contact rate (Test 2). The error is computed as the maximum 1-norm difference between the marginal distributions from each ODE solver and the FRM.

ODE solver	Time cost and note	Error
Adams-PECE	243s	0.0550
Runge-Kutta	172s	0.0550
BDF-GM-LU0	189s	136.2503
BDF-LU0	17s	71.4624
MAGNUS-SSA-1	3924s	0.0550
MAGNUS-SSA-2	2211s	0.0550
MAGNUS-SSA-3	2165s	0.0572
MAGNUS-SSA-4	509s	0.0552

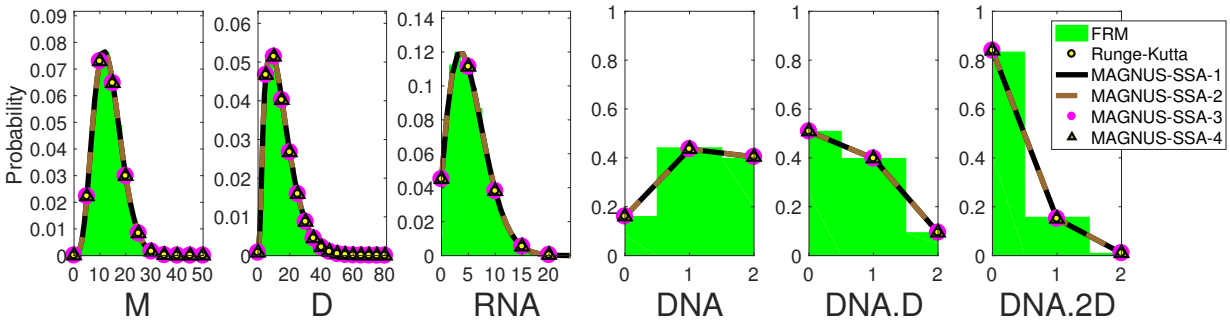


Figure 5.5: Probability distributions from the transcriptional regulatory model (Test 3)

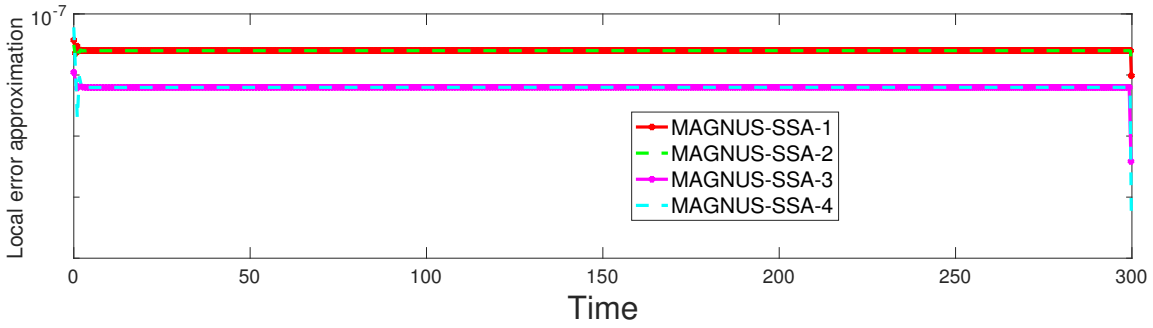


Figure 5.6: Error estimates in the **MAGNUS-SSA** variants from the transcriptional regulatory model (Test 3)

5.6.3 Test 3 - the transcriptional regulatory model

The final biological problem for comparing the ODE solvers depicts a transcriptional regulatory system [54]. The problem consists of six species:

$$\begin{aligned}
\text{M} &: \text{protein (monomer)},\\
\text{D} &: \text{transcription factor (dimer)},\\
\text{DNA} &: \text{DNA template, free of dimers},\\
\text{DNA.D} &: \text{DNA template, bound at one binding site},\\
\text{DNA.2D} &: \text{DNA template, bound at both binding sites},\\
\text{RNA} &: \text{mRNA produced by transcription},
\end{aligned}$$

which can interact through ten reactions:

$$\begin{aligned}
\text{RNA} &\xrightarrow{c_1} \text{RNA} + \text{M},\\
\text{M} &\xrightarrow{c_2} \varnothing,\\
\text{DNA.D} &\xrightarrow{c_3} \text{RNA} + \text{DNA.D},\\
\text{RNA} &\xrightarrow{c_4} \varnothing,\\
\text{DNA} + \text{D} &\xrightarrow{c_5} \text{DNA.D},\\
\text{DNA.D} &\xrightarrow{c_6} \text{DNA} + \text{D},\\
\text{DNA.D} + \text{D} &\xrightarrow{c_7} \text{DNA.2D},\\
\text{DNA.2D} &\xrightarrow{c_8} \text{DNA.D} + \text{D},\\
\text{M} + \text{M} &\xrightarrow{c_9} \text{D},\\
\text{D} &\xrightarrow{c_{10}} \text{M} + \text{M}.
\end{aligned}$$

The reaction rates are:

$$
\begin{aligned}
c_1 &= 0.043 \ (\mathrm{s}^{-1}), \\
c_2 &= 0.0007 \ (\mathrm{s}^{-1}), \\
c_3 &= 0.078 \ (\mathrm{s}^{-1}), \\
c_4 &= 0.0039 \ (\mathrm{s}^{-1}), \\
c_5 &= \frac{0.012 \cdot 10^9}{A \cdot V(t)} \ (\mathrm{s}^{-1}), \\
c_6 &= 0.4791 \ (\mathrm{s}^{-1}), \\
c_7 &= \frac{0.00012 \cdot 10^9}{A \cdot V(t)} \ (\mathrm{s}^{-1}), \\
c_8 &= 0.8765 \cdot 10^{-11} \ (\mathrm{s}^{-1}), \\
c_9 &= \frac{0.05 \cdot 10^9}{A \cdot V(t)} \ (\mathrm{s}^{-1}), \\
c_{10} &= 0.5 \ (\mathrm{s}^{-1}),
\end{aligned}
$$

where A is the Avogado's constant, and $V(t)$ is the cell volume at time t, which increases from the initial value $V(0) = 10^{-15}$ in accordance to

$$V(t) = V(0)e^{\ln(2)t/\tau}$$

during the entire cell cycle time period $\tau = 35$ minutes until the cell divides.

We wish to follow the distributions of the count of each species from the initial state where the cell has 2 monomers, 6 dimers, and two unbound DNAs:

$$
\begin{aligned}
\mathrm{M} &= 2, & \mathrm{D} &= 6, \\
\mathrm{DNA} &= 2, & \mathrm{DNA.D} &= 0, \\
\mathrm{DNA.2D} &= 0, & \mathrm{RNA} &= 0,
\end{aligned}
$$

until the final time $t_f = 300$ (s).

Table 5.3: Results from the ODE solvers for the transcriptional regulatory model (Test 3). The error is computed as the maximum 1-norm difference between the marginal distributions from each ODE solver and the FRM.

ODE solver	Time cost and note	Error
Adams-PECE	Incorrect solver (flag 5)	N/A
Runge-Kutta	82415s	0.0465
BDF-GM-LU0	1203s	78.2631
BDF-LU0	88s	273.3214
MAGNUS-SSA-1	137304s	0.0465
MAGNUS-SSA-2	52764s	0.0465
MAGNUS-SSA-3	78505s	0.0465
MAGNUS-SSA-4	49139s	0.0465

The FSP bounds are

$$\begin{aligned} 0 \leq \quad \text{M} \quad \leq 50, \quad 0 \leq \quad \text{D} \quad \leq 81, \\ 0 \leq \quad \text{DNA} \quad \leq 2, \quad 0 \leq \quad \text{DNA.D} \quad \leq 2, \\ 0 \leq \quad \text{DNA.2D} \quad \leq 2, \quad 0 \leq \quad \text{RNA} \quad \leq 24. \end{aligned}$$

There are $n = 2822850$ states in the FSP state space, and the CME matrix contains $nz = 24093072$ nonzero elements. The infinity norm of the CME matrix at $t = 0$ is 311973491.

Table 5.3 summarizes the results from the ODE solvers. The probability distributions from the ODE solvers and the error estimates are plotted in figures 5.5 and 5.6. Similarly to Test 2, **BDF-GM-LU0** and **BDF-LU0** produce inaccurate results (see Table 5.3). Furthermore, **Adams-PECE** detected that the problem is stiff and therefore did not finish. Their solutions are therefore not included in Figure 5.5. **Runge-Kutta** and all **MAGNUS-SSA** variants finished and their solutions followed the FRM frequencies faithfully. Figure 5.6 shows that the **MAGNUS-SSA** error estimates are stable, with clear agreement between the 2nd-order and the 4th-order approximations, respectively.

5.6.4 Observations

The three biological problems in the numerical tests were chosen because they exhibit distinct features. The model of two competing T cell clonotypes is symmetrical. Over time, the probability mass concentrates on states with low V_1 counts and high V_2 counts, or those with high V_1 counts and low V_2 counts, because the model is also bimodal. The epidemic model with periodic contact rate, on the other hand, is monomodal with periodic time-dependency. Finally, the transcriptional regulatory model is monomodal and very stiff, evidenced in the differences in magnitude of the reaction rates and the big norm of the CME matrix.

The initial conditions for the problems in the three numerical tests were chosen so that they are biologically relevant. Specifically, the initial state in Test 3 is from the original problem [54]. The initial condition for Test 1 follows the same structure as in [81], and the initial values for Test 2 is the same as in [56], except the lower value for S, which is dictated by storage requirements. We also performed tests for the same biological problems with slightly different initial conditions, and found the results to be qualitatively similar to those presented here. This means that the algorithms are sensitive to a similar extent to the initial conditions.

BDF-GM-LU0 and **BDF-LU0** finished in all three numerical tests. They escape the storage requirement of the complete LU decomposition by applying a fast and cheap incomplete LU decomposition instead where, in the course of the decomposition, the elements not in the sparsity pattern of $\boldsymbol{A}(t)$ are discarded. The price for the low storage requirement and cheap computation time cost is that the incomplete LU decomposition might lose valuable information during the integration time, evidenced by the large error of the marginal distributions. Because of this, the probability distributions computed by **BDF-GM-LU0** and **BDF-LU0** are wrong in all three tests. Tables 5.1, 5.2 and 5.3 confirm that their results always have the largest errors. For instance, in Test 1, which is the smallest problem, they fail to follow the shape of the marginal probability

distributions. This is in contrast to results in [86], in which **BDF-GM-LU0** and **BDF-LU0** performed better than their complete LU variants across a range of ODE problems. This is because for those problems, the matrices are sparse with small norms, so the elements discarded by the incomplete LU decomposition are indeed insignificant. Performing the complete LU decomposition in that case would be time-consuming without substantially improved accuracy.

Adams-PECE did not finish in Test 1, because the maximum number of steps allowed in the program was exceeded before reaching t_f. In Test 2, it produced correct distributions in competitive time, only surpassed by **Runge-Kutta** (**RK**). In Test 3 **Adams-PECE** did not finish because it detected that the problem is stiff.

Quite surprisingly, **RK** was the only solver besides the **MAGNUS-SSA** algorithms to complete for all three numerical tests. This is interesting, given that among the traditional ODE solvers in this comparison, it is the only explicit algorithm. Even more, **RK** finished first in Tests 1 and 2, because its explicit formula requires less computation time. **RK** produced the correct probability distribution in Test 3, while the other traditional ODE solvers failed because the problem is stiff. Note that in our previous report [87], all ODE solvers were tested for the transcriptional regulatory model with the same set-up as Test 3, but with initial values

$$\begin{aligned} \mathrm{M} &= 0, & \mathrm{D} &= 2, \\ \mathrm{DNA} &= 1, & \mathrm{DNA.D} &= 0, \\ \mathrm{DNA.2D} &= 0, & \mathrm{RNA} &= 0, \end{aligned}$$

final time $t_f = 600s$, FSP bounds

$$\begin{aligned} 0 \leq \mathrm{M} &\leq 46, & 0 \leq \mathrm{D} &\leq 59, \\ 0 \leq \mathrm{DNA} &\leq 1, & 0 \leq \mathrm{DNA.D} &\leq 1, \\ 0 \leq \mathrm{DNA.2D} &\leq 1, & 0 \leq \mathrm{RNA} &\leq 12, \end{aligned}$$

and $n = 293280, nz = 2091993$. **RK** did not finish in that test, possibly because of the longer end time.

All **MAGNUS-SSA** algorithms produced exact probability distributions in all numerical tests, proving that Magnus-based integration methods are suitable for solving CME problems with time-dependent rates. On the other hand, **MAGNUS-SSA** is time-consuming, because there is at least one exponential-matrix-vector product to be computed every time step. Therefore all **MAGNUS-SSA** implementations were exceeded by **RK** in Test 1, and **RK** and **Adams-PECE** in Test 2. However, all **MAGNUS-SSA** implementations except for **MAGNUS-SSA-4** outpaced **RK** in Test 3, because the Magnus-based methods have been known to excel in solving highly oscillatory or stiff problems [72].

We now compare the different **MAGNUS-SSA** algorithms. They all produced good results in the tests, and it is not surprising that **MAGNUS-SSA-1** always finished last. It requires two Expokit runs per time step, which is time-consuming, and the error estimate is of order 2, implying the time steps are taken conservatively. **MAGNUS-SSA-2** employs our cheap approximation of the local error estimate used in **MAGNUS-SSA-1**, and it is faster in all numerical tests, even more than halving the time cost in Test 3. Figures 5.2, 5.4 and 5.6 reveal that the error estimates in **MAGNUS-SSA-2** follow closely those in **MAGNUS-SSA-1**, explaining why its solutions are very dependable.

MAGNUS-SSA-3 applies the common halved time-step scheme. Because both the normal Magnus solution and its 'corrector' are of the 4th order, so is its error. This, in combination with the fact that its error estimates tend to be lower than those in **MAGNUS-SSA-1** and **MAGNUS-SSA-2**, result in more relaxed time steps and ultimately small time costs, even though there are three Expokit runs for every time step.

MAGNUS-SSA-4 is the fastest among the Magnus implementations in all tests, and there are several reasons for this. Firstly, and most importantly, its error estimate is a

priori, allowing the algorithm to check the time step before committing to the time-demanding task of finding the next probability distribution. Moreover, there is only one Expokit run for each time step. It is possible that for the problems where $\boldsymbol{A}(t)$ changes dramatically, **MAGNUS-SSA-4** might fail, because its error estimate does not take the quadrature error into account. However, many biological problems with time-dependent rates are modeled using continuous functions for the reaction rates, in which case **MAGNUS-SSA-4** is a good candidate as an ODE solver.

5.7 Conclusions

In this chapter, we developed the framework of Magnus-SSA, which allows the FSP state space to be adaptively changed during the Magnus integration time. This decreases the time costs of the Magnus-based methods. We also implemented two local error approximating schemes from MacNamara and Burrage [81], and Iserles, Marthinsen and Nørsett [71], as well as two other error estimation techniques. The resulting Magnus-SSA algorithms were then tested against Adams-PECE, Runge-Kutta, and BDF, across several problems in biology where the CME arises. The numerical results show that Magnus-based methods are serious candidates for solving stiff CME problems with time-dependent rates.

It is important to point out that Adams-PECE, Runge-Kutta, and BDF are 'all-purpose' ODE solvers, where the Magnus-based methods in consideration here are only suitable for solving the linear ODE in the form of (5.1). However, such problems arise in many biological fields and even in other sciences where Markovian reaction networks occur [4]. Numerical Magnus-based methods, therefore, can be an important choice for solving these problems. On the other hand, there is already interest in expanding the Magnus method to solving non-autonomous linear ODE problems [60]. This may be of interest to the biology community, where such problems can emerge [88].

CHAPTER 6

APPLICATION OF THE KRYLOV-FSP-SSA METHOD IN PARAMETER INFERENCE

This chapter is based on work published in Physical Biology [89].

6.1 Introduction

In this chapter, we will consider an important application for the numerical and theoretical methods that we have developed so far. One of the important goals of systems biology is to understand the complex and stochastic dynamics of gene regulation. A challenge toward this goal is that there are usually many unknown reaction rates in the involved mathematical models.

We use a data-driven maximum likelihood approach [90, 91, 92, 93, 94, 95, 96] to search for and validate unknown parameters so that the distributions reported in the experimental data can be recreated in the models. We apply this approach to synthetic data generated from the negative feedback model in Min Wu et al. [3], where an inhibitory gene network was constructed using two synthetic promoters [97]. Their lab experiment involved TetR and LacI (Fig. 6.1), which are repressors that inhibit the expression of each other by binding to their corresponding operator sites, TetR operator (Otet) and LacI operator (Olac), placed in engineered GAL1 promoters. Anhydrotetracycline (ATc) was used to inhibit TetR. The abundance of each protein was recorded by flow cytometry with yeast-enhanced green fluorescent protein (yEGFP) and mCherry red fluorescent protein.

A mathematical model consisting of a set of two ordinary differential equations (ODEs) was proposed to explain the interaction of the two proteins involved [3, 97]. Experiments in [3] showed bimodality that is disruptive to the ODE model. This is because the dynamics of cellular processes with low copy numbers of molecules can be

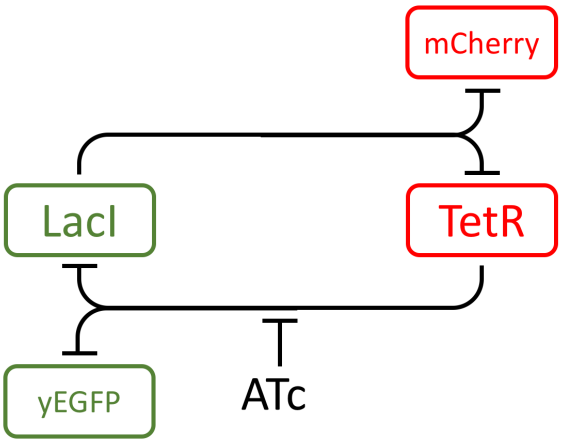


Figure 6.1: Schematic diagram of the network used in Min Wu et al. [3].

noisy events [98, 99, 100] and so the deterministic ODE formulation is not always ideal. This is supported by real time measurements of RNAs and proteins using fluoresent proteins made possible by recent advances in bio-imaging [101, 102, 103, 104, 105]. Because of this, stochastic models have arisen as a natural modeling choice in many cases in systems biology [106, 107, 92]. The problem of integrating stochastic models with single-cell data is, therefore, of relevance to the systems biology community [108]. But in order to use either deterministic or stochastic biological models for analysis or for designing future lab experiments with confidence, unknown parameters need to be found so that the models can capture the qualitative and quantitative features of the distributions found in the data [2, 109].

Parameter fitting in stochastic models is harder [91, 93, 92], although it can offer important insights, particularly in systems biology where fitting of statistics or distributions can reveal some information about the underlying biological parameters or mechanisms. For example in [110], the probability distribution of nascent RNA was obtained by a stochastic model. It predicted hitherto unobserved discontinuities and periodic peaks in the distribution, which were then verified experimentally. Aslo in [13], parameter identification and cross-validation analyses were employed to choose among many stochastic model hypotheses the best model that fits the data without losing its

predictive power because of overfitting. There have been other works that gained valuable insights from using stochastic models [99, 100]. Moreover, while rate constants derived from deterministic parameter values have been used in many published CME models, the noise in the system can generate dynamics that are different from the predictions of deterministic models [91, 111]. Using deterministic parameters in a stochastic model can thus be deceptive. Hence, parameter inference in stochastic models is a relevant problem.

This has kindled the interest of several recent efforts aimed at facilitating the task [91, 112, 111, 109]. In this study, we use the concept of maximum likelihood [90, 91, 92, 93, 94, 95, 96], which has been regarded as a natural approach given the probabilistic nature of stochastic models [108]. The general principle of maximum likelihood parameter estimation [96, 13, 110] is to find the parameters with which the mathematical model can reproduce the distributions in the experiments by using the likelihood of the data given a parameter set as the objective function for an optimization problem. Hence, fitting parameters in a stochastic model using maximum likelihood is essentially an optimization problem. What makes this problem challenging is that there can be confounded parameters, an identifiability issue or multimodality of the likelihood surface [108]. Although there have been comparisons of different derivative-free optimization schemes [113, 114], there is no single optimization scheme that performs best across all test problems [113]. Because of this, it is important to compare different optimization algorithms in the specific context of parameter fitting using the maximum likelihood. Here, we compare the performances of five optimization algorithms (three local and two global), representing some of the popular optimization techniques.

The rest of the chapter is organized as follows: Section 6.2 forms the CME for the particular gene regulation case under consideration. The likelihood of experimental observations given some parameter set is defined in Section 6.3, as well as the parameter fitting scheme as an optimization process. The likelihood function is found by solving the CME, which is formidable and further compounded with the many function evaluations

required for the optimization problem. Section 6.4 outlines the Krylov-FSP-SSA algorithm, which is a powerful numerical component that we use for this purpose. We report some numerical tests in Section 6.5, followed by a discussion and some concluding remarks in Section 6.6.

6.2 The CME for the TetR-LacI gene regulation problem

The application considered in this study is found in Min Wu et al. [3, 97], where their mathematical model uses a set of two ODEs to characterize the interaction of the two proteins:

$$\frac{d[\text{LacI}]}{dt} = c_{rl} + p_{e,tet} \cdot (c_{il} - c_{rl}) - \delta \cdot [\text{LacI}] \tag{6.1}$$

$$\frac{d[\text{TetR}]}{dt} = c_{rt} + p_{e,lac} \cdot (c_{it} - c_{rt}) - \delta \cdot [\text{TetR}] \tag{6.2}$$

We will detail the variables $p_{e,tet}$, $p_{e,lac}$, and constants c_{il}, c_{rl}, c_{it}, c_{rt}, δ when describing the CME. We will see that there is a total of 11 parameters, 6 of which are estimated from previous experiments and the remaining 5 are to be fitted.

To describe the stochastic alternative based on the CME, we define the *state vector* consisting of 2 proteins species: TetR and LacI, and represented as

$$\mathbf{x} = ([\text{TetR}], [\text{LacI}])^T \tag{6.3}$$

where [TetR] and [LacI] can be any nonnegative integer counting the corresponding proteins. We model the interaction between the two species using the following reactions:

$$\varnothing \xrightarrow{\kappa_1} \text{LacI} \tag{6.4}$$

$$\text{LacI} \xrightarrow{\kappa_2} \varnothing \tag{6.5}$$

$$\varnothing \xrightarrow{\kappa_3} \text{TetR} \tag{6.6}$$

$$\text{TetR} \xrightarrow{\kappa_4} \varnothing, \tag{6.7}$$

where κ_i is the reaction rate of reaction i. The formulae for these rates can be found in [3] and are summarized below.

The quantity $p_{e,tet}$ is the probability of TX (the promoter for LacI) to not be bound by TetR. Given the state vector at the moment, it can be defined as

$$K_I = k_{ATc} \cdot [\text{TetR}] \tag{6.8}$$

$$f_I = \left(\frac{K_I}{K_I + [ATc] \cdot k_t} \right)^m \tag{6.9}$$

$$p_{e,tet} = \frac{k_t^{n_t}}{k_t^{n_t} + ([\text{TetR}] \cdot f_I)^{n_t}} \tag{6.10}$$

where the parameter k_{ATc}, nonlinearity constant n_t, and k_t (defined to be the active [TetR] needed so that $p_{e,tet} = 50\%$) are to be fitted. Note that from fitting the Hill coefficient of induction of ATc to the dose response curves, we have

$$m \cdot n_t = 11.5 \tag{6.11}$$

and therefore only need to find n_t.

If TX is not bound by TetR (with probability $p_{e,tet}$), the production rate of LacI is c_{il} (min^{-1}). However, if TetR does not bind to TX (with probability $1 - p_{e,tet}$), the production rate of LacI is c_{rl} (min^{-1}). Therefore we have:

$$\kappa_1 = p_{e,tet} \cdot c_{il} + (1 - p_{e,tet}) \cdot c_{rl} \tag{6.12}$$

$$= c_{rl} + p_{e,tet} \cdot (c_{il} - c_{rl}) \tag{6.13}$$

Similarly, $p_{e,lac}$ is defined as the probability of LX (the promoter for TetR) to not be bound by LacI, and is given as

$$p_{e,lac} = \frac{k_l^{n_l}}{k_l^{n_l} + [\text{LacI}]^{n_l}} \tag{6.14}$$

where the nonlinearity constant n_l and parameter k_l (active [LacI] needed so that $p_{e,lac} = 50\%$) are unknown. We then have

$$\kappa_3 = c_{rt} + p_{e,lac} \cdot (c_{it} - c_{rt}) \tag{6.15}$$

where c_{rt} (min^{-1}) and c_{it} (min^{-1}) are TetR production rates when LX is repressed or induced, respectively.

Finally, because TetR and LacI are both very stable proteins, the decrease of intracellular abundance of these repressors is through cell division. Yeast cells grown in galactose media have doubling times of about 6 hours [3, 97], corresponding to

$$\kappa_2 = \kappa_4 = \delta \approx 0.002\ (min^{-1}). \tag{6.16}$$

We can then define the propensities of the four reactions given the state of the system, which are the probabilities of them occurring during an infinitesimal time interval $[t, t + dt)$:

$$\alpha_1 = c_{rl} + p_{e,tet} \cdot (c_{il} - c_{rl}) \tag{6.17}$$

$$= c_{rl} + \frac{k_t^{n_t} \cdot (c_{il} - c_{rl})}{k_t^{n_t} + \left[[\text{TetR}] \cdot \left(\frac{k_{ATc} \cdot [\text{TetR}]}{k_{ATc} \cdot [\text{TetR}] + [ATc] \cdot k_t}\right)^m\right]^{n_t}} \tag{6.18}$$

$$\alpha_2 = \delta \cdot [\text{LacI}] \tag{6.19}$$

$$\alpha_3 = c_{rt} + p_{e,lac} \cdot (c_{it} - c_{rt}) \tag{6.20}$$

$$= c_{rt} + \frac{k_l^{n_l} \cdot (c_{it} - c_{rt})}{k_l^{n_l} + [\text{LacI}]^{n_l}} \tag{6.21}$$

$$\alpha_4 = \delta \cdot [\text{TetR}] \tag{6.22}$$

where the parameters k_{ATc}, k_t and k_l, and nonlinearity constants n_t and n_l are unknown and need to be fitted by the experimental data (note that since these are dimensionless quantities used to calculate $p_{e,tet}$ and $p_{e,lac}$, they do not require any unit).

The other parameters are fixed and come from experiments with promoters [97]:

$$c_{rl} = 7.46 \ (min^{-1}) \tag{6.23}$$

$$c_{il} = 918 \ (min^{-1}) \tag{6.24}$$

$$c_{rt} = 13.06 \ (min^{-1}) \tag{6.25}$$

$$c_{it} = 717.38 \ (min^{-1}) \tag{6.26}$$

$$m = \frac{11.5}{n_t} \tag{6.27}$$

where the last equation comes from (6.11).

The CME [5] for this particular problem becomes:

$$\frac{dP(\mathbf{x},t)}{dt} = \sum_{k=1}^{4} \alpha_k(\mathbf{x} - \boldsymbol{\nu}_k)P(\mathbf{x} - \boldsymbol{\nu}_k, t) - \sum_{k=1}^{4} \alpha_k(\mathbf{x})P(\mathbf{x}, t) \tag{6.28}$$

where the stoichiometric vector $\boldsymbol{\nu}_k$ represents the change in species numbers if reaction k occurs. The ODE form of equation (6.28) for the n possible states $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is then:

$$\dot{\mathbf{p}}(t) = \mathbf{A} \cdot \mathbf{p}(t) \tag{6.29}$$

where $\mathbf{p} = (P(\mathbf{x}_1, t), \ldots, P(\mathbf{x}_n, t))^T$ and the transition rate matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$ is defined as

$$a_{ij} = \begin{cases} -\sum_{k=1}^{4} \alpha_k(\mathbf{x}_j), \text{ if } i = j \\ \alpha_k(\mathbf{x}_j), \text{ if } \mathbf{x}_i = \mathbf{x}_j + \boldsymbol{\nu}_k \\ 0, \text{ otherwise} \end{cases} \tag{6.30}$$

The agreement between the dynamics described by the CME and the original ODEs [3] will be shown in Section 6.5.

6.3 Parameter fitting

Parameter inference in stochastic biochemical systems has been less developed than in deterministic models [93]. Maximum likelihood [96, 13, 110] represents a natural approach for this problem because of the probabilistic nature of stochastic models. In this section we will define the likelihood function for the specific experimental data of Min Wu et al. [3], but to help make the contrast with our approach clear, we first briefly describe their approach to parameter fitting using the ODE model.

There are two sets of lab experiments in [3]. In the first experiment, TXLX[1] cultures are treated with full ATc induction (250 ng/mL) for 48hr. In the second experiment, the cultures are treated with no ATc induction during the same time frame. Both cultures are then rediluted into media containing different ATc levels and the yEGFP measurements are recorded. The normalized GFP plots, as seen in Fig. 1D-F in [3] for the two experiments, do not coincide, which indicates bimodality. The ATc region where the plots are distinct is called the bistable region [3].

The goal of the fitting scheme in their work [3] is to find the parameters (k_{ATc}, n_t, k_t, n_l and k_k) so that the bistable region predicted by the mathematical model fits the experimental data. A range for each parameter is specified, so that they have biologically reasonable values. Random parameter sets are then generated uniformly from these regions. The bistable region for each set is then calculated, and only those whose bistable regions are within 10% relative error from the experimentally established region are kept.

We explore here a more general approach to fitting, in which the goal is not to fit the bistable region but to find the parameter set so that the frequencies shown in the experimental data can be captured in the mathematical model. Since in general, the flow cytometry measurements performed at different time points for different experiments produce histograms of the protein numbers, the goal of our approach of parameter fitting is to calibrate the parameters, which are k_{ATc}, n_t, k_t, n_l and k_k in this application, so that the

[1] In [3], the gene network constructed using the TX and LX synthetic promoters is called TXLX.

probability distributions predicted by the mathematical model at these time points fit the experimental results. This can be formulated using the definition of likelihood function.

Suppose that N cells were under observation, and the ith cell was measured at time point t_i of experiment e_i and found to belong to the state $\mathbf{x}_i = ([\text{TetR}]_i, [\text{LacI}]_i)^T$. Assuming a parameter set $\theta = (k_{ATc}, n_t, k_t, n_l, k_k)^T$, we can solve the CME to compute the probability that a given cell is in that state, which is $p(\mathbf{x}_i|\theta, e_i, t_i)$. The total likelihood of all observations, $L(\mathbf{D}|\theta)$, is the product of the probabilities of all observed cells:

$$L(\mathbf{D}|\theta) = \prod_{i=1}^{N} p(\mathbf{x}_i|\theta, e_i, t_i). \tag{6.31}$$

The problem of parameter fitting is then to find the parameter set θ_{Fit} that maximizes this likelihood, or equivalently, the logarithm of the likelihood:

$$\theta_{\text{Fit}} = \arg\max_\theta(L(\mathbf{D}|\theta)) \tag{6.32}$$

$$= \arg\max_\theta(\log(L(\mathbf{D}|\theta))) \tag{6.33}$$

$$= \arg\max_\theta(\sum_{n=1}^{N} \log(p(\mathbf{x}_i|\theta, e_i, t_i))). \tag{6.34}$$

An optimization routine is required to find θ_{Fit}. To conduct parameter searches, we employ five different optimization algorithms: PRAXIS [115], NELMIN [116, 117] and NEWUOA [118, 119], representing local optimization approaches, together with GLOBAL [120, 121, 122] and SIMANN [123, 124] which are two global optimization algorithms.

We note that there have been other works that dealt with modeling and analyzing experimental data of a genetic toggle switch, some of them very similar to the model discussed here [90, 2, 109] but with differences in the parameter fitting schemes. In [2] and [109], Munsky and Khammash fitted single-cell data using statistical quantities, such as the mean levels, marginal distributions, or full joint distributions, employing the FSP to compute the solutions to the CME. The fitted parameter arguments are then chosen to

minimize the difference between the measured statistical quantity and the numerical solution of that quantity, using the 1-norm since the FSP naturally provides exact bounds on the 1-norm error of the solution. Their search was run using multiple iterations of fminsearch in MATLAB, which is a local optimization algorithm, and a simulated annealing algorithm. In this work, we use the concept of maximum likelihood to fit the parameters instead.

6.4 The Krylov-FSP-SSA algorithm

In the maximum likelihood approach, the CME is repeatedly solved over a large number of parameter sets, from which the likelihood of each parameter set can be computed and the parameters with the maximal likelihood is chosen. As we have seen in previous chapters, solving the CME is a formidable task. There are infinitely many states that the system can occupy when the copy numbers of species in the system are not bounded (as in the problem of interest here). The choice of the CME solver is therefore crucial to the effectiveness of this approach. The Stochastic Simulation Algorithm (SSA) or other Monte Carlo methods were chosen in many maximum likelihood works [91, 92, 93, 94, 95]. They avoid the curse of dimensionality by drawing random trajectories of the system and using the resulting frequencies to indirectly approximate the true probability distributions. In this chapter, we employ the FSP to directly solve the CME.

An advantage of solving the CME directly by the FSP, detailed in previous sections, is that unlike Monte Carlo methods, such as the SSA [6, 7, 8] or its many improved variants [9, 16, 17, 125, 126, 127], the FSP possesses an analytical bound on the error of the resulting probability distributions. As the number of states taken into account in the FSP is increased, this bound is decreased and the probability of any given state of the system is more accurate. This contrasts with Monte Carlo methods, where the error is statistical.

Among existing implementations mentioned in Chapter 2, the Krylov-FSP-SSA method by Sidje and Vo [1] turns out to be reliable and efficient. The method uses SSA

trajectories to find the likely states during the interval $[t_k, t_k + \tau_k]$ of small length τ_k, and Krylov techniques [37] for evaluating the matrix exponential, which are among the most effective strategies, especially when the matrix is large and sparse [82]. Since the state space is kept compact, containing only the most likely states of the system, $\mathbf{A}_{J_k}$ is usually considerably smaller than it would be in the original FSP algorithm, and therefore the time taken by the matrix exponential is even further reduced.

We performed a trial comparison between the Krylov-FSP-SSA and another FSP implementation [2]. We observed that, across 100 evaluations with randomized parameter sets, they achieved comparable accuracy but the former took on average 23s while the latter took 134s (Section 6.5 has more details). It should be noted that there are many different FSP implementations, and choosing the right algorithm for any specific problem is not always obvious. However, the Krylov-FSP-SSA algorithm proved to be a powerful enough tool for our task that involves repeated solves of the CME during the optimization process.

6.5 Numerical tests

6.5.1 Computing platform

All tests reported here utilized resources of the Alabama Supercomputer Authority, which at the time of writing houses two supercomputers called SGI UV and DMC. The user can request a job to be executed on either of them, or can simply let the operating system select the more suitable system depending on the workload and availability. All codes were written in FORTRAN 77 and were run on the large queue of the SGI UV with 1 processor core (Xeon E5-4640 CPU operating at 2.4 GHz), 360hr time limit and 1GB memory limit.

6.5.2 Comparison between the CME and ODE models

In Section 6.2, we rewrote the deterministic ODE system in [3] into the stochastic CME. Therefore it is important to show that the two models indeed describe the same evolution in protein counts, which will be confirmed now.

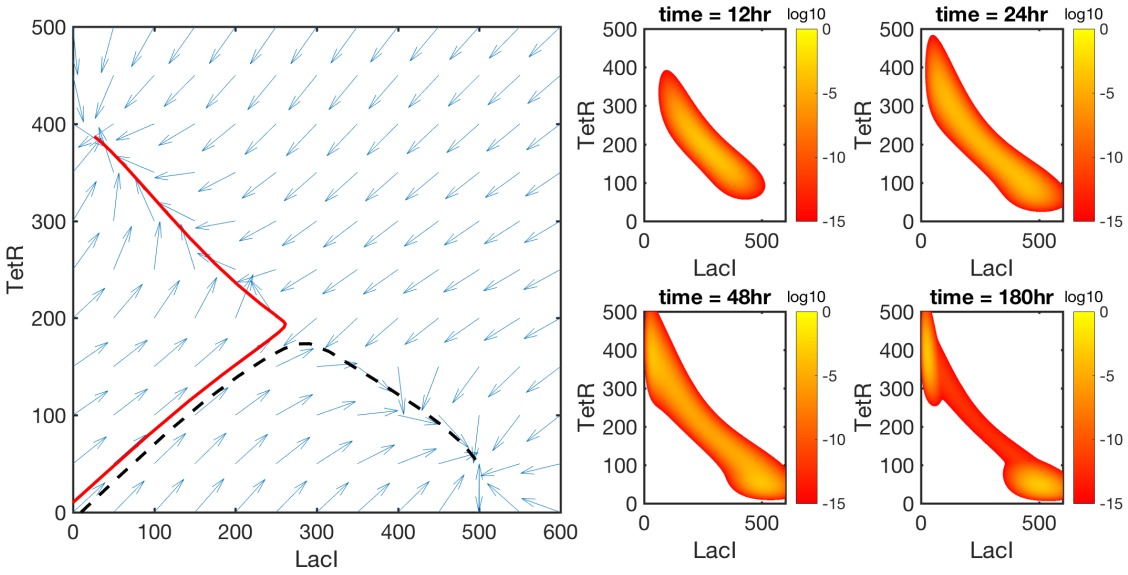


Figure 6.2: Left panel: vector field of the ODEs [3] when the parameters are $k_{ATc} = 0.94, k_t = 11, n_t = 1.56, k_l = 264, n_l = 3.35$. The red solid line and the black dashed line represent two solutions of the ODEs when the initial state is $([\text{TetR}], [\text{LacI}]) = (10, 0)$ and $(0, 10)$, respectively. Four panels on the right: evolution of the probability distribution at time 12, 24, 48 and 180hr from solving the CME with the same parameter set.

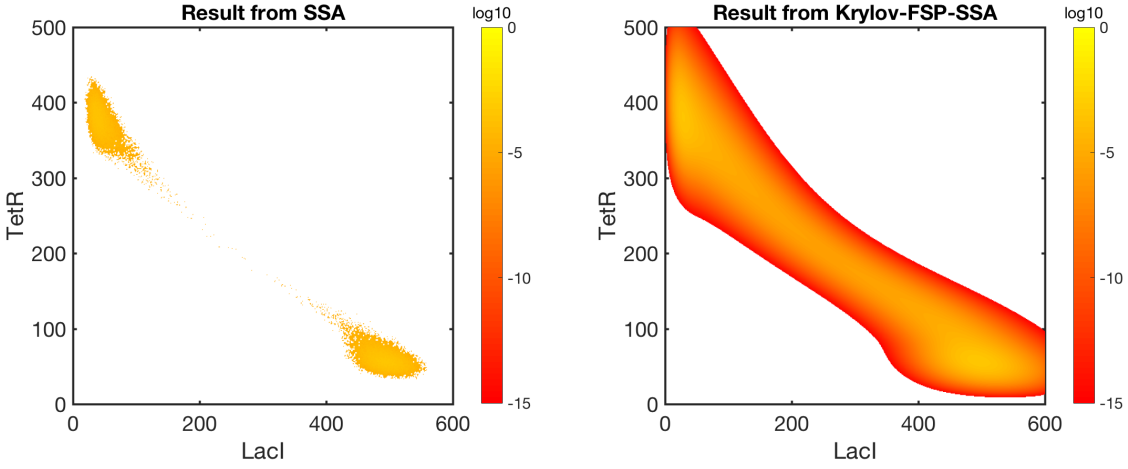


Figure 6.3: The probability distribution at 48hr, computed by 100,000 SSA runs (left plot) and by the Krylov-FSP-SSA algorithm (right plot). The parameters are $k_{ATc} = 0.94, k_t = 11, n_t = 1.56, k_l = 264, n_l = 3.35$.

Figure 6.2 shows the vector field of the ODE model, which dictates the evolution in numbers of TetR and LacI in any cell. The ODEs are solved for two different initial states:

$$([\text{TetR}], [\text{LacI}]) = (0, 10) \tag{6.35}$$

and

$$([\text{TetR}], [\text{LacI}]) = (10, 0) \tag{6.36}$$

and the solutions are superimposed on the vector field (the black dashed line and the red solid line, respectively). Even though the initial states are close to each other, the trajectories branch out to two different steady states. This demonstrates the stochasticity of the system, where a small change in the protein counts at the beginning can lead to two different cell fates.

The CME is then solved with the Krylov-FSP-SSA algorithm for the same parameter set and the resulting transient probability distributions are also shown in Figure 6.2. As predicted in the vector field of the ODEs, the probability mass first drifts toward the unstable steady state, then it is divided between the two modes of the bimodal distribution.

There are several observations to be drawn from this numerical test. First of all, the fact that the distributions resulting from the CME agree with the vector field of the ODEs implies that the two models describe the same problem, confirming the reliability of the CME. Second, even though the ODEs' vector field can predict both the unstable and stable states, it cannot produce the transient distributions which are required for computing the likelihood function. These transient distributions also give a clearer picture of the cell's fate. For example, solving the ODEs with initial state $([\text{TetR}], [\text{LacI}]) = (0, 0)$ only results in one steady state. However, the stochastic nature of the system implies that the system might end up in the second steady state instead. The CME not only predicts that but also shows the probability for the cell to commit to either outcome. This is

difficult to do using the ODE model.

6.5.3 Comparison between Krylov-FSP-SSA and SSA

Having checked that the CME model is compatible with the original ODE model in [3], we will now assess the choice of the Krylov-FSP-SSA as the CME solver for computing the likelihood function instead of Monte Carlo methods such as the SSA.

Figure 6.3 compares the probability distribution when the parameters are:

$$k_{ATc} = 0.94, k_t = 11, n_t = 1.56, k_l = 264, n_l = 3.35 \tag{6.37}$$

at 48hr, when the system is in equilibrium, by using on one hand the Krylov-FSP-SSA algorithm, and on the other hand 100,000 trajectories of the classic SSA, which took 2 minutes in runtime. A clear contrast here is that solving the CME by the Krylov-FSP-SSA algorithm took about 10s. Recall also the advantage of FSP algorithms to offer computations accurate to an a priori threshold (set here to be 10^{-5}).

In previous works of maximum likelihood estimation for parameter inference in stochastic models [90, 91, 92, 93, 94, 95], the SSA or other Monte Carlo approaches were used for computing the likelihood function. However, the large number of different parameter sets to be examined means that it is not realistic to compute many realizations for each parameter set. The resulting distributions might therefore be incomplete. By contrast, for small gene regulation problems, where the state space is small enough and the integration interval is not too long, FSP algorithms can supply the probabilities for many more states in a reasonable short runtime.

6.5.4 Comparison between Krylov-FSP-SSA and original FSP

There has not been a systematic comparison between different FSP implementations in a wide range of biological problems. The Krylov-FSP-SSA algorithm was chosen here for several reasons. First, it can be used without prior assumptions on the model. Several other FSP algorithms require bounds on the state space, and this cannot

be known without trial runs to find the areas on the state space that accumulate the most probability mass. The Krylov-FSP-SSA algorithm, however, finds the state space on the fly by following the direction of a few SSA runs and therefore does not need a priori bounds on the protein counts. Second, with its time-stepping feature, the Krylov-FSP-SSA algorithm can be more efficient than other FSP algorithms.

To check its effectiveness for the model under consideration in this study, we compare it to a FSP implementation by Munsky [2] with the parameter set (6.37). Munsky's FSP implementation was translated from its original MATLAB code to FORTRAN for a fair comparison, since it is well-known that FORTRAN is magnitudes faster than MATLAB.

The two algorithms were tested for 100 different parameter sets in the parameter space. Each parameter set is randomly picked from the uniform distribution in its range, chosen to be the same as that used in [3]:

$$0.01 < k_{ATc} < 1 \tag{6.38}$$

$$1 < k_t < 400 \tag{6.39}$$

$$1 < n_t < 5 \tag{6.40}$$

$$1 < k_l < 400 \tag{6.41}$$

$$1 < n_l < 5 \tag{6.42}$$

For each parameter set, we record the runtime for finding the probability distributions by either algorithm and then computing the likelihood function based on the distributions. The average runtime of each algorithm is shown in Table 6.1. It is also crucial to check that the two algorithms give the same likelihood value. For this, we

Table 6.1: Comparison between the Krylov-FSP-SSA [1] and Munsky's FSP implementation [2] using 100 evaluations with randomized parameter sets to compute the averages.

Average runtime of the Krylov-FSP-SSA	23s
Average runtime of the FSP in [2]	134s
Average relative 1-norm error (6.43)	1.23×10^{-2}

compute the relative 1-norm error for each parameter set θ:

$$relerr = \frac{|L_1(\mathbf{D}|\theta) - L_2(\mathbf{D}|\theta)|}{|L_2(\mathbf{D}|\theta)|} \tag{6.43}$$

where $L_1(\mathbf{D}|\theta)$ is the likelihood computed by the Krylov-FSP-SSA algorithm, and $L_2(\mathbf{D}|\theta)$ is computed by Munsky's FSP implementation. The average relative 1-norm error of the 100 parameter sets is shown in Table 6.1.

We can clearly see that even though the resulting likelihoods are practically the same between the two algorithms, the Krylov-FSP-SSA has a much shorter average runtime. This is because there are a number of stark differences between them, notably the fact that the Krylov-FSP-SSA is a time-stepping algorithm, unlike Munsky's implementation in [2].

It is important to note that there are other FSP variants, many of which are time-stepping [12, 11, 19, 128] and some might be more efficient than the Krylov-FSP-SSA in some instances. However, there has not yet been an in-depth numerical comparison between the variants, and the Krylov-FSP-SSA was retained because of its availability and its satisfactory performance for our job.

With the CME solver chosen, the final piece of the puzzle is to pick an optimization algorithm for finding the parameter set with maximum likelihood. There are many different derivative-free optimization schemes and their variants. There have also been works to compare these schemes in a variety of test problems, e.g., [113, 114]. Overall, the performance of the optimization schemes depends on the specific problems, and there is no single optimization scheme that is guaranteed to perform best in all circumstances [113].

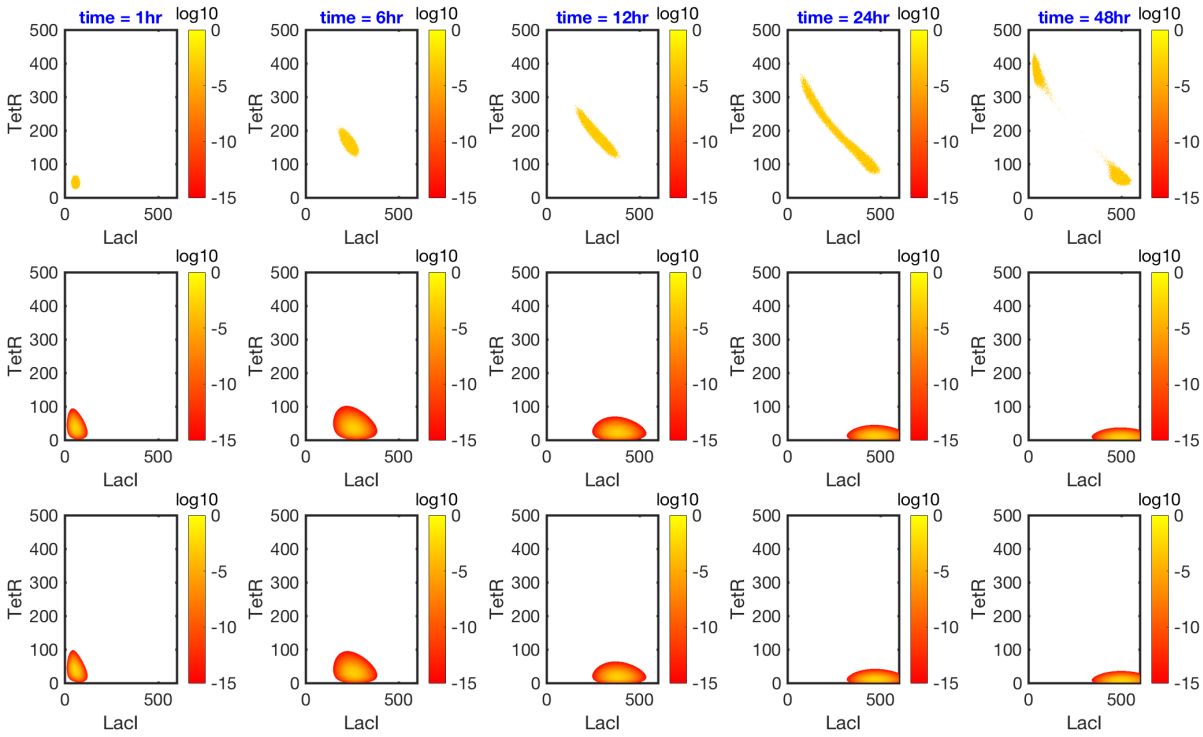


Figure 6.4: (Test 1) Result of the optimization algorithms with starting parameter guess: $k_{ATc} = 0.2, k_t = 250, n_t = 4, k_l = 55, n_l = 3$. First row: the synthetic data consisting of protein counts for 100,000 cells per time point, computed by SSA with the true parameters $k_{ATc} = 0.94, k_t = 11, n_t = 1.56, k_l = 264, n_l = 3.35$. Second row: the distributions at corresponding time points with the starting parameter guess. Third row: the distributions at corresponding time points with the final parameter guess.

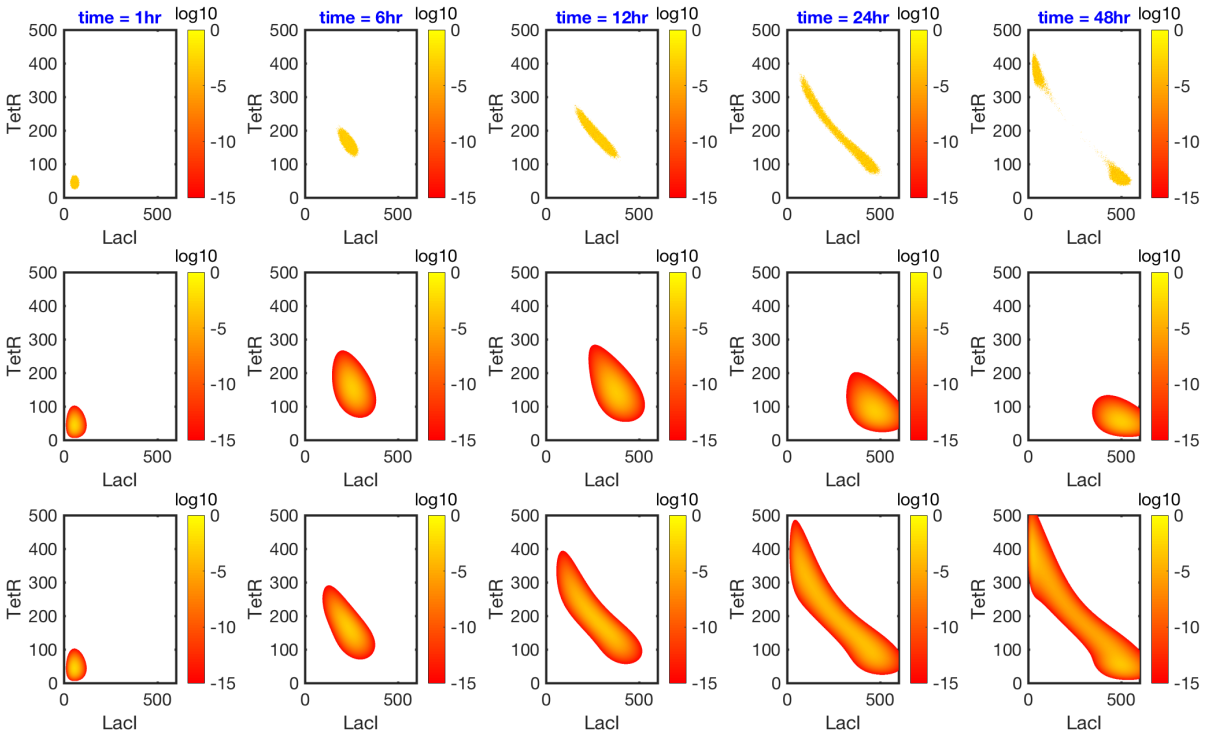


Figure 6.5: (Test 2) Result of the optimization algorithms with starting parameter guess: $k_{ATc} = 0.9, k_t = 13, n_t = 1, k_l = 255, n_l = 3$. First row: the synthetic data consisting of protein counts for 100,000 cells per time point, computed by SSA with the true parameters $k_{ATc} = 0.94, k_t = 11, n_t = 1.56, k_l = 264, n_l = 3.35$. Second row: the distributions at corresponding time points with the starting parameter guess. Third row: the distributions at corresponding time points with the final parameter guess.

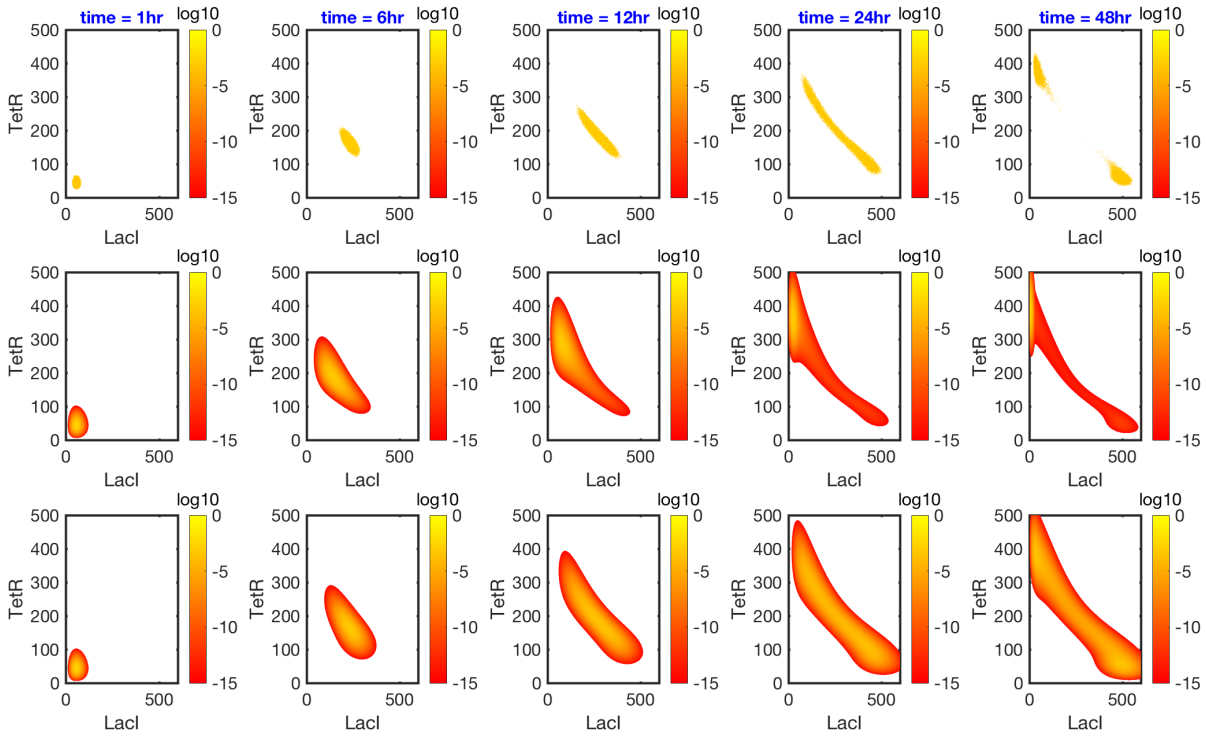


Figure 6.6: (Test 3) Result of the optimization algorithms with starting parameter guess: $k_{ATc} = 0.8, k_t = 8, n_t = 2, k_l = 280, n_l = 4$. First row: the synthetic data consisting of protein counts for 100,000 cells per time point, computed by SSA with the true parameters $k_{ATc} = 0.94, k_t = 11, n_t = 1.56, k_l = 264, n_l = 3.35$. Second row: the distributions at corresponding time points with the starting parameter guess. Third row: the distributions at corresponding time points with the final parameter guess.

Table 6.2: Input parameters of the local optimization algorithms

	Input parameters	Value	Definition
PRAXIS	T_0	10^{-3}	Error tolerance
	MACHEP	2.22×10^{-16}	Machine precision
	H_0	0.1	Maximum stepsize
	FMIN	10^6	Estimate of minimum (used only for printing)
NELMIN	STEP	$[0.1, 0.1, 1, 1, 0.05]$	Size and shape of the initial simplex
	REQMIN	10^6	Terminating limit for the variance of the function values
	KONVGE	10	Convergence check
	KCOUNT	1000	Maximum number of function evaluations
NEWUOA	NPT	11	Number of interpolation conditions
	RHOBEG	0.8	Initial value of a trust region radius
	RHOEND	1	Final value of a trust region radius
	MAXFUN	1000	Maximum number of function evaluations

Therefore, in this study we compare some popular local and global optimization algorithms for the specific task of stochastic parameter fitting with maximum likelihood. All of them are readily available online in FORTRAN and represent different optimization approaches.

Some of the optimization routines being investigated are maximization schemes, while others are minimization schemes. In the latter case, the sign of the likelihood function is simply reversed. Also, the routines originally written using the single precision REAL data type have all been changed to DOUBLE PRECISION for a fair comparison.

6.5.5 Local optimization schemes

We include three different local optimization algorithms for the fitting scheme:

- PRAXIS [115], one of the first derivative-free optimization solvers developed, using Powell's method of conjugate search directions

- NELMIN [116, 117], an implementation of the Nelder-Mead algorithm for

Table 6.3: Results of the local optimization algorithms

		n_t	n_l	k_t	k_l	k_{ATc}	Number of function evalutations	Likelihood
Synthetic data		1.56	3.35	11	264	0.94		-1873172.8
Test 1	Initial guess	4	3	250	55	0.20		-6329761.9
	PRAXIS	4.02	5.00	249.93	55.01	0.08	131 (33 linear searches)	-6314107.0
	NELMIN	2.26	4.98	245.05	76.39	0.02	193	-6294412.4
	NEWUOA	4.00	3.80	250.00	55.00	0.60	14	-6321248.2
Test 2	Initial guess	1	3	13	255	0.90		-3659419.6
	PRAXIS	1.87	3.31	13.33	265.27	1.00	268 (83 linear searches)	-1884653.9
	NELMIN	1.25	3.05	9.17	255.02	0.97	1007	-1923230.3
	NEWUOA	1.20	3.00	13.00	255.00	0.90	13	-3549849.8
Test 3	Initial guess	2	4	8	280	0.80		-5413880.6
	PRAXIS	1.47	3.35	8.24	264.32	0.68	346 (121 linear searches)	-1873192.5
	NELMIN	1.65	3.55	12.14	277.00	0.99	1004	-1905593.2
	NEWUOA	2.00	4.00	8.00	280.00	0.40	13	-2037143.6

derivative-free optimization

- NEWUOA [118, 119], implementing Powell's model-based algorithm using trust regions.

Aside from an initial guess for the parameter set, these routines require the input parameters shown in Table 6.2. These values are recommended in the codes and are therefore used here.

As will be shown later, the performance of these optimization algorithms depends heavily on the starting guess: if the initial guess for the parameters is too far away from the correct parameters, the algorithms are less likely to provide a good output. The experimental data [3] consists of the protein numbers at different time points. To compare the performances of the optimization codes, we produce *synthetic data* by solving for the distribution vectors resulting from the CME with the initial state $([\text{TetR}], [\text{LacI}]) = (0, 0)$ with the true parameter set (6.37) at 5 different time points: 1hr, 6hr, 12hr, 24hr, and 48hr. For each time point, 100,000 samples are randomly drawn from the distribution vectors.

The frequencies of the protein counts at the 5 time points are the input for the fitting scheme and the goal is to find the parameter set that can recreate the distribution in the synthetic data. We remark here that synthetic data allows us to easily test our fitting method and the performance of the algorithms on a variety of input, knowing that the results on the synthetic data are indicative of the results to be expected on experimental data. There is a limit of maximum 1000 function evaluations, which is adequate for these algorithms to converge to some maxima.

The biological model [3], as is the case with most models, defines specific ranges for the parameters, which were given in (6.38)–(6.42). During the process, if the parameter set proposed from the optimization scheme is out of this range, its likelihood is defined to be a very small number (-10^{-21}) to dissuade the scheme from traveling in this direction.

To highlight how the initial guess plays a crucial role when using a local optimization scheme, we implement all three schemes from three different initial guesses.

In the first test:

$$k_{ATc} = 0.2, k_t = 250, n_t = 4, k_l = 55, n_l = 3 \tag{6.44}$$

in the second test:

$$k_{ATc} = 0.9, k_t = 13, n_t = 1, k_l = 255, n_l = 3 \tag{6.45}$$

and finally in the third test:

$$k_{ATc} = 0.8, k_t = 8, n_t = 2, k_l = 280, n_l = 4 \tag{6.46}$$

The numbers of function evaluations that each scheme requires, the final optimal parameter set and its likelihood are shown in Table 6.3.

As can be seen in Table 6.3, the initial guess for the first test is very far from the true parameter set used to produce the synthetic data, resulting in a small likelihood. The final results from the local optimization schemes from this initial guess, therefore, only slightly improve the likelihood. The distributions they produce are similar and shown in Figure 6.4. These distributions fail to recreate the distributions observed in the synthetic data, as expected.

The initial guess for the second test is closer to the true parameter set, and the final results from PRAXIS and NELMIN reflect this: their optimal likelihoods are greatly increased from the initial likelihood, and are very close to the true value. Between these two algorithms, the result from PRAXIS is better (larger final likelihood) and the algorithm converges after a much smaller number of function evaluations. On the other hand, NEWUOA converges after only 13 iterations and only marginally increases the likelihood. The distributions resulting from PRAXIS and NELMIN are shown in Figure 6.5. As reflected in the large likelihood, the final result recreates the bimodality in the synthetic data and the evolution of probability distribution over time. Even though the initial guess produces distributions that are very different from the data, the optimization codes can easily calibrate the parameters to maximize the likelihood function and arrive at

the distributions almost identical to the synthetic data.

In the third test, the small likelihood of the initial guess indicates that it is far from the optimal point. Nevertheless, the results from PRAXIS and NELMIN are still very close to the true values. Similar to the second test, PRAXIS converges after less function evaluations to a better solution than NELMIN, and NEWUOA converges after only 13 iterations to the worst solution among the three algorithms. Figure 6.6 shows that the initial parameter guess produces distributions very different from the synthetic data, but the final result from the fitting scheme is similar to the frequency shown in the synthetic data.

We can conclude from these three tests that the performance of local optimization algorithms depends greatly on the initial guess. This reflects the fact that the likelihood function is difficult to optimize: it is likely not convex, and has many local maxima that these algorithms cannot escape from. Among the three algorithms, PRAXIS produces better results than NELMIN in spite of much smaller number of function evaluations needed for it to converge. NEWUOA converges quickly but its results are inferior to PRAXIS and NELMIN.

In real life applications, however, neither the true parameter set nor its likelihood are known in advance. This therefore casts doubt on employing these local optimization algorithms. A solution to this might be to start the local optimization scheme from different initial guesses, randomly chosen from the range, and select the best final result. This is the strategy of a number of global optimization algorithms, including GlobalSearch and MultiStart in MATLAB [129, 130]. An advantage of this is that each optimization run is independent from the others, and so it is possible to produce a parallel algorithm. Another strategy is to employ global optimization schemes. These algorithms focus on finding the maximum over the entire range and will be investigated in the next section.

6.5.6 Global optimization schemes

Two global optimization algorithms are investigated:

Table 6.4: Input parameters of the global optimization algorithms

	Input parameters	Value	Definition
GLOBAL	NSIG	6	Convergence criterion
	M	1	Number of residual functions
	N100	500	Number of sample points to be drawn uniformly in one cycle
	NG0	10	Number of best points selected from the actual sample
	SEED	$[1, 2, 3, 4, 5, 6]$	Seeds for the random number generator
SIMANN	T_0		Initial temperature
	X	[4, 3, 250, 55, 0.2]	Initial guess for the parameter set
	RT	0.85	Temperature reduction factor
	EPS	10	Error tolerance for termination
	NS	20	Number of cycles
	NT	100	Number of iterations before temperature reduction
	NEPS	4	Number of final function values to decide upon termination
	MAXEVL	5000	Maximum number of function evaluations
	C	[2, 2, 2, 2, 2]	Vector controlling the step length adjustment
	ISEED1	1	First seed for the random number generator
	ISEED2	2	Second seed for the random number generator
	VM	[1, 1, 1, 1, 1]	Step length vector

Table 6.5: Results of the global optimization algorithms

	n_t	n_l	k_t	k_l	k_{ATc}	Number of function evaluations	Likelihood
Synthetic data	1.56	3.35	11	264	0.94		-1873172.8
GLOBAL	1.50	3.35	9.24	264.00	0.77	4572	-1873171.5
	1.55	3.35	10.63	264.01	0.90		-1873172.2
	1.29	3.35	4.31	263.98	0.33		-1873181.5
	3.40	3.17	21.41	267.17	1.00		-2002408.4
	3.54	2.95	8.16	269.13	0.22		-2078322.3
	4.35	2.32	22.18	283.77	0.83		-2321101.8
	4.53	4.34	18.74	329.37	0.47		-2637311.5
	2.14	1.88	7.67	310.07	0.38		-2668259.4
	3.07	2.11	19.95	324.27	0.99		-2743173.7
	4.15	2.65	18.98	342.64	0.59		-2916667.9
	3.51	2.35	2.94	346.36	0.05		-3004177.6
	3.32	2.91	381.46	268.37	0.02		-3740211.5
	3.08	3.02	172.91	272.32	0.76		-3744919.8
	3.75	2.76	29.49	264.93	0.03		-3745244.7
	4.64	3.10	73.69	266.65	0.37		-3746429.0
	3.22	3.01	270.00	261.40	0.22		-3748192.9
	1.51	3.21	163.64	266.39	0.47		-3750487.6
	2.90	2.64	283.87	262.79	0.59		-3757378.6
	1.61	3.40	283.71	281.80	0.95		-3762507.3
	4.60	3.47	334.80	280.11	0.69		-3767111.1
SIMANN from $T_0 = 10^1$	2.64	1.00	247.53	81.58	0.59	limit exceeded	-5375590.2
SIMANN from $T_0 = 10^2$	4.01	1.00	248.27	72.83	0.23	limit exceeded	-5495580.3
SIMANN from $T_0 = 10^3$	3.17	1.00	246.87	77.95	0.40	limit exceeded	-5425506.7
SIMANN from $T_0 = 10^4$	1.46	3.63	9.76	269.04	0.83	limit exceeded	-1873618.3
SIMANN from $T_0 = 10^5$	2.54	4.27	15.10	271.02	0.77	limit exceeded	-1880111.8
SIMANN from $T_0 = 10^6$	3.86	4.96	213.19	284.03	0.78	limit exceeded	-2142633.2
SIMANN from $T_0 = 10^7$	4.35	3.90	31.67	257.66	0.51	limit exceeded	-2034590.3
SIMANN from $T_0 = 10^8$	1.86	1.53	80.26	254.45	0.81	limit exceeded	-2146360.6
SIMANN from $T_0 = 10^9$	4.04	1.90	374.31	75.31	0.88	limit exceeded	-2041629.5
SIMANN from $T_0 = 10^{10}$	2.64	2.52	180.20	342.90	0.46	limit exceeded	-2086480.4

- GLOBAL [120, 121, 122], based on the Boender-Rinnooy-Stougie-Timmer algorithm [120, 121] and is a stochastic method involving sampling, clustering and local search. It was implemented in FORTRAN by Csendes [122], and the output contains up to 20 local maxima.

- SIMANN [123, 124], a simulated annealing algorithm [131, 132, 133, 134].

The input parameters required by these routines are shown in Table 6.4. Similarly to the tests with local optimization schemes, the values used here are recommended by the codes when available.

The comparison of these two algorithms uses the same synthetic data in the previous section as the data for fitting parameters, as well as the range for the parameters. A limit of 5000 function evaluations is applied on each algorithm. In practice this has been shown to be adequate for good results.

Unlike GLOBAL, the algorithm SIMANN as well as other simulated annealing algorithms depend on important input parameters to produce good results. The algorithm escapes from local optima, which is important to produce better results than local optimization schemes, by accepting downhill steps. This decision is made by the Metropolis criteria using T ("temperature") and the downhill move size in a probabilistic way. The downhill move is more likely to be accepted if T and the move size are smaller.

Therefore, the importance of the parameter T in the performance of SIMANN cannot be overstated. A smaller initial T_0 might result in a step length too small, and the function evaluations gathered by the algorithm are not enough to find the optima. The choice of an optimal initial temperature T_0, however, depends on the problem and trial runs usually have to be performed in order to find the right T_0. Because of this, we performed ten different tests with SIMANN, each with a different initial temperature:

$$T_0 = 10^k, k = 1, \ldots, 10 \tag{6.47}$$

SIMANN also requires an initial guess, and in our tests this is chosen to be

$$k_{ATc} = 0.2, k_t = 250, n_t = 4, k_l = 55, n_l = 3 \tag{6.48}$$

Note that this initial guess was chosen for Test 1 in the previous section and was shown to result in unsatisfactory solutions from the local optimization schemes.

The results from GLOBAL and SIMANN, with different initial temperatures, are shown in Table 6.5.

The GLOBAL algorithm finished after 4572 function evaluations. By default, it outputs 20 local maxima found during the process, shown in Table 6.5 with decreasing likelihoods. On the other hand, the result notes that there are too many clusters, implying that there are a large number of local maxima, confirming the reason for the failure of local optimization schemes: since the likelihood surface is multimodal, they converge to the nearest local maximum and cannot escape, which is why the results depend on the initial guesses.

Despite this, GLOBAL was able to find very good results for the parameter set. The first three results have virtually the same likelihoods as the true parameter set. The parameters themselves are also very close to the true values, except k_{ATc}, which might imply that the likelihood function is not very sensitive to this parameter.

On the other hand, all SIMANN runs exceeded the limit of 5000 function evaluations, which might be a result of the tight error tolerance (the variable EPS in Table 6.4). As expected, the SIMANN runs starting with small initial temperature ($T_0 < 10^4$) result in parameter sets with very small likelihoods. When $T_0 = 10^4$, SIMANN converges to a good result, with likelihood only slightly smaller than that of the true parameter sets. With $T_0 > 10^4$, however, the results from SIMANN are worse.

It is important to point out, however, that even with the optimal initial temperature of $T_0 = 10^4$, the parameter set from SIMANN is not as good as the three best

results from GLOBAL. The latter also outputs the local maxima, which are important to draw conclusions about the likelihood function itself, as opposed to only one best parameter set as in SIMANN, and it does all this with less function evaluations. Importantly, the input parameters that GLOBAL requires are not significant to the final result, while SIMANN depends on some important input parameters which can only be set up by experience, knowledge of the problem, or trial optimization runs.

Although GLOBAL seems to be superior to SIMANN for this problem, the same conclusion cannot be drawn universally. GLOBAL in particular, and the Boender-Rinnooy-Stougie-Timmer algorithm in general, will first sample parameter sets in the given range, and then transform the parameter sets into groups around local maxima. Clustering techniques are then employed to find neighborhoods of each local maximum, and local optimization runs from each cluster can point to the global maximum. While effective for problems with few parameters, other optimization algorithms can be more effective when there are hundreds or more parameters to be found.

6.5.7 Sensitivity effect

Finally, we investigate how sensitive the likelihood function of the synthetic data is with respect to each parameter in Figure 6.7. The true parameter set (6.37) was used to generate the synthetic data, and the changes in the likelihood function when each parameter varies around its true value (with the other four parameters fixed to their exact numbers) are shown.

The likelihood function does not change much when k_l or n_l vary in their neighborhoods. In comparison, the likelihood responds more strongly to changes in k_t or k_{ATC}. When n_t increases from 1 to 1.56, however, the likelihood roughly increases two-fold in its value, implying that the model is most sensitive to this parameter. This sensitivity study may have important ramifications especially in parameter fitting, since not knowing the sensitivity of the parameters may lead to the codes spending time calibrating the non-sensitive parameters without getting a good result.

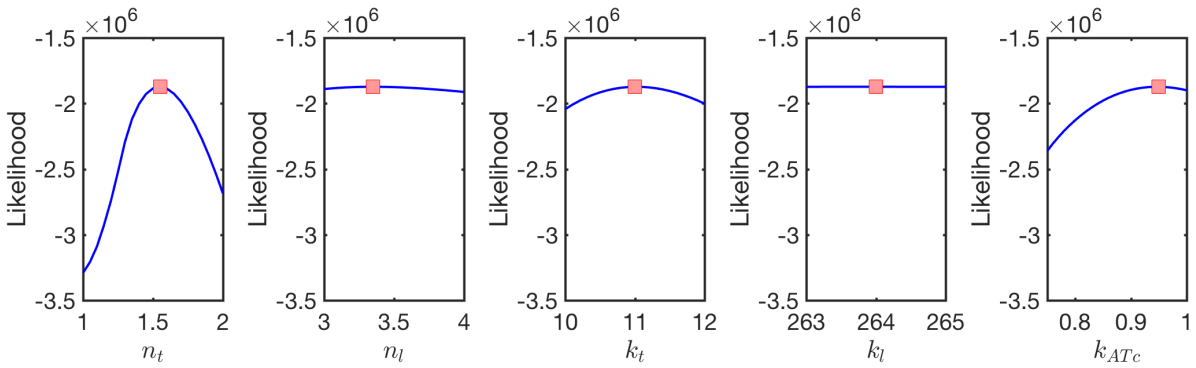


Figure 6.7: The sensitivity of the likelihood function of the synthetic data with respect to each parameter around the parameter set $k_{ATc} = 0.94, k_t = 11, n_t = 1.56, k_l = 264, n_l = 3.35$. The panels show the change in the likelihood function when there is a change in n_t, n_l, k_t, k_l and k_{ATc}, respectively. The red squares correspond to the likelihood when the correct parameters are used.

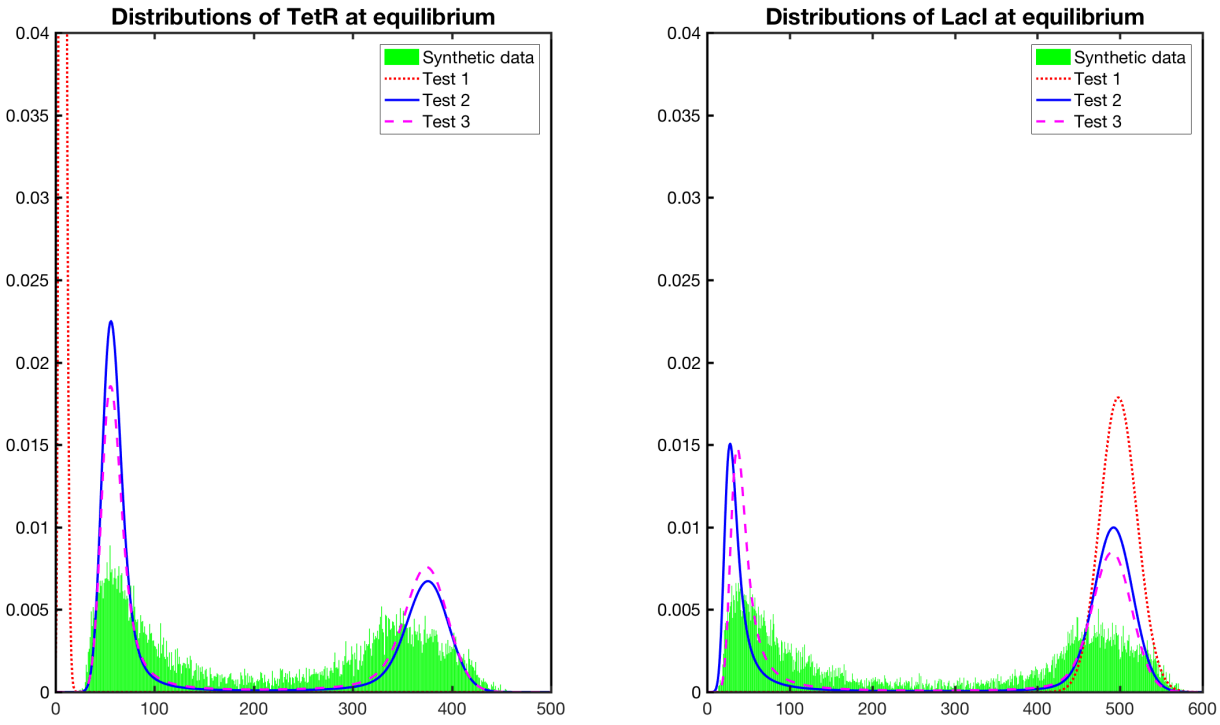


Figure 6.8: Comparison of the distributions at equilibrium (180hr) between the results from the different tests and the synthetic data that they are fitting.

6.6 Conclusion

Synthetic biology is an effective approach to study how microbes and multicellular organisms regulate their cell fate determination when the environments change or proper development needs to be ensured. An investigation into the different properties of the network requires a mathematical model that illuminates possible regulatory mechanisms, for which a systematic approach to calibrating free parameters is required. The model can then be cross-validated using other datasets or used to predict/refine outcomes of new experiments.

Here we have investigated a mutual inhibitory gene network in Saccharomyces cerevisiae, using the model from [3, 97]. The likelihood function is evaluated by using the Krylov-FSP-SSA algorithm to solve the CME for the probability distributions given a parameter set. The likelihood function is then maximized by different optimization codes. This ensures that the solution results in a faithful portrayal of the evolution of the probability distribution over time and therefore confirms the mathematical model.

We compared for the first time different optimization algorithms for parameter fitting using maximum likelihood. There is still work to be done, as only one biological problem is considered here. There are also many different optimization approaches, and a more complete comparison between them for a variety of stochastic models will be beneficial to the systems biology community, given the importance of parameter inference in this field. The results in this work might be one step further towards establishing a parameter inference method of reference in stochastic models.

From the numerical tests in Section 6.5, it is apparent that local optimization schemes do not perform well for our purpose. This underlines the difficulty of fitting stochastic models by maximum likelihood. The likelihood surface is often multimodal, and therefore it is difficult for the local optimization algorithms to escape a local maxima.

Figure 6.8 offers a comparison of the marginal distributions for both proteins from the three tests with local optimization algorithms at 180hr, where the system reaches

equilibrium and clearly shows a bimodal pattern. Since the results from PRAXIS and NELMIN do not differ much in the resulted distributions even though the parameter sets are not the same, we only use the results from PRAXIS. The marginal distributions from test 1 do not match the data, with the TetR distribution being inconsistent with the synthetic data and the LacI distibution showing unimodality instead of bimodality. On the other hand, tests 2 and 3 show a bimodality in agreement with the synthetic data, although there is a slight difference in the height of the peaks of the probability distribution. Note that each peak represents one mode, or one possible fate that the cell can end up in.

In practice, choosing a starting parameter set for each local optimization run can be a challenging task. Usually, a range for each parameter is chosen so that they are biologically relevant. The parameter search is then conducted by randomly choosing different initial guesses for the parameters in these ranges, leading to thousands of function evaluations per optimization run, for which only the best solution is recorded at the end. This task can be done in an embarrassingly parallel code, since the optimization runs are independent from each other. As can be seen from Table 6.3, the results are often satisfactory when the starting parameter guess is good. It is thus possible to have satisfactory results by employing the local optimization schemes in a parallel multi-start fashion.

On the other hand, the numerical comparison showed that global optimization algorithms produce better results than local optimization schemes, at the expense of more function evaluations. Only two global optimization schemes were considered in this study, of which GLOBAL [120, 121, 122] proved to be the better choice. This of course might change when a different biological model is tested, as GLOBAL is effective only for problems with few unknown parameters.

Our comparison considered only optimization algorithms in FORTRAN that are freely available. There have been other works comparing local and global optimization methods [113, 114] but the test cases in those works belong to different classes from that

studied here. A broader and deeper comparison involving more biological models and more optimization algorithms might be essential to the biomathematical community given the importance of the parameter fitting problem.

CHAPTER 7

CONCLUSIONS

The chemical master equation (CME) is a popular approach for modeling the stochastic interactions within a biological cell, and the finite state projection (FSP) is an efficient technique to solve it. This dissertation has reviewed many FSP variants for approximating the CME solution when the reaction rates are constant, as well as investigated novel methods for the CME with time-varying rates. We also established the theoretical background for the FSP through the framework of inexact Krylov methods, and examined applications in delay CME and parameter inference involving local and global optimization schemes.

New questions arise from chapters 5 and 6. First, parallelism has not been considered in the parameter inference. It can be effective in the parameter search which has to be conducted by randomly choosing different initial guesses for the parameters, leading to thousands of function evaluations per optimization run, for which only the best solution is recorded at the end. This task can be done in an embarrassingly parallel code, since the optimization runs are independent from each other, allowing a parallel multi-start search. Second, the sensitivity of the model parameters in the likelihood function needs to be explored in the context of solving the CME. This might prove helpful in developing new parameter fitting schemes for stochastic biological models. Third, even though a Magnus-based integration method has been developed in chapter 5 and shown to be efficient for solving certain stiff problems, more numerical comparisons must be made to confirm this phenomenon. It might also be possible to parallelize the algorithm in order to fully leverage the power of modern supercomputers. For instance, the SSA runs performed to expand the state space can run in parallel as well, which would reduce the runtime.

Another promising target for parallelization is the computation of the Krylov subspace. The Magnus-based algorithm can also be combined with the tensor format, which has been successfully applied for solving a CME with constant reaction rates [128].

REFERENCES

[1] R. B. Sidje and H. Vo. Solving the chemical master equation by a fast adaptive finite state projection based on the stochastic simulation algorithm. *Mathematical Biosciences*, 269:10–16, 2015.

[2] B. Munsky. *Quantitative biology from molecular to cellular systems*, chapter Modeling cellular variability, pages 234–266. Taylor and Francis group, New York, 2012.

[3] M. Wu, R. Su, X. Li, T. Ellis, Y.-C. Lai, and X. Wang. Engineering of regulated stochastic cell fate determination. *PNAS*, 110(26):10610–10615, 2013.

[4] J. Goutsias and G. Jenkinson. Markovian dynamics on complex reaction networks. *Phys. Rep.*, 529(2):199–264, 2013.

[5] D. T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 188(1-3):404–425, 1992.

[6] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.*, 22(4):403–434, 1976.

[7] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 115(4):1716–1733, 2001.

[8] Y. Cao, D. T. Gillespie, and L. R. Petzold. Efficient step size selection for the tau-leaping simulation method. *J. Chem. Phys.*, 124(4):044109, 2006.

[9] Y. Cao, D. T. Gillespie, and L. R. Petzold. The slow-scale stochastic simulation algorithm. *J. Chem. Phys.*, 122(1):14116, 2005.

[10] B. Munsky and M. Khammash. The finite state projection algorithm for the solution of the chemical master equation. *J. Chem. Phys.*, 124(4):044104, 2006.

[11] K. Burrage, M. Hegland, S. Macnamara, and R. B. Sidje. A Krylov-based finite state projection algorithm for solving the chemical master equation arising in the discrete modelling of biological systems. In *Markov Anniversary Meeting: an international conference to celebrate the 150th anniversary of the birth of A.A. Markov*, pages 1–18, 2006.

[12] B. Munsky and M. Khammash. A multiple time interval finite state projection algorithm for the solution to the chemical master equation. *J. Comput. Phys.*, 226(1):818–835, 2007.

[13] G. Neuert, B. Munsky, R. Z. Tan, L. Teytelman, M. Khammash, and A. v. Oudenaarden. Systematic identification of signal-activated stochastic gene regulation. *Science*, 339(6119):584–587, 2013.

[14] K. N. Dinh and R. B. Sidje. Understanding the finite state projection and related methods for solving the chemical master equation. *Physical biology*, 13(035003), 2016.

[15] T. Jahnke and W. Huisinga. Solving the chemical master equation for monomolecular reaction systems analytically. *J. Math. Biol.*, 54(1):1–26, 2007.

[16] Y. Cao, D. T. Gillespie, and L. R. Petzold. Avoiding negative populations in explicit Poisson tau-leaping. *J. Chem. Phys.*, 123:054104, 2005.

[17] A. Chatterjee, D. G. Vlachos, and M. A. Katsoulakis. Binomial distribution based tau-leap accelerated stochastic simulation. *J. Chem. Phys.*, 122(2):24112, 2005.

[18] B. Munsky. *Modeling cellular variability*, chapter 11, pages 233–266. Taylor and Francis, Inc, 2011.

[19] V. Wolf, R. Goel, M. Mateescu, and T. A. Henzinger. Solving the chemical master equation using sliding windows. *BMC Syst. Biol.*, 4:42, 2010.

[20] V. Sunkara and M. Hegland. An optimal finite state projection method. *Procedia Comput. Sci.*, 1(1):1579–1586, 2010.

[21] V. Sunkara. *Analysis and numerics of the Chemical Master Equation.* PhD thesis, Australian National University, 2013.

[22] S. Peles, B. Munsky, and M. Khammash. Reduction and solution of the chemical master equation using time scale separation and finite state projection. *J. Chem. Phys.*, 125(20):204104, 2006.

[23] M. Hegland, C. Burden, L. Santoso, S. MacNamara, and H. Booth. A solver for the stochastic master equation applied to gene regulatory networks. *J. Comput. Appl. Math.*, 205:708–724, 2007.

[24] J. J. Tapia, J. R. Faeder, and B. Munsky. Adaptive coarse-graining for transient and quasi-equilibrium analyses of stochastic gene regulation. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 5361–5366, 2012.

[25] R. B. Sidje, K. Burrage, and S. MacNamara. Inexact uniformization method for computing transient distributions of Markov chains. *SIAM J. Sci. Comput.*, 29(6):2562–2580, 2007.

[26] W. K. Grassmann. Transient solutions in markovian queueing systems. *Comput. Oper. Res.*, 4(1):47–53, 1977.

[27] D. Gross and D. R. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Oper. Res.*, 32(2):343, 1982.

[28] M. Hegland and J. Garcke. On the numerical solution of the chemical master equation with sums of rank one tensors. In *ANZIAM J.*, volume 52, pages C628–C643, 2011.

[29] V. Wolf. Modelling of Biochemical Reactions by Stochastic Automata Networks. *Electron. Notes Theor. Comput. Sci.*, 171(2):197–208, 2007.

[30] V. Kazeev, M. Khammash, M. Nip, and C. Schwab. Direct solution of the Chemical Master Equation using quantized tensor trains. *PLoS Comput. Biol.*, 10(3), 2014.

[31] S. Dolgov and B. N. Khoromskij. Tensor-product approach to global time-space-parametric discretization of chemical master equation. *Preprint 68, Max-Planck-Institut für Mathematik in den Naturwissenschaften*, 2012.

[32] A. Cichocki. Era of big data processing: a new approach via tensor networks and tensor decompositions. *arXiv*, pages 1–30, 2014.

[33] S. Dolgov and B. Khoromskij. Simultaneous state-time approximation of the chemical master equation using tensor product formats. *arXiv*, 2013.

[34] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2008.

[35] S. MacNamara, K. Burrage, and R. B. Sidje. Multiscale modeling of chemical kinetics via the Master Equation. *Multiscale Model. Simul.*, 6(4):1146–1168, 2008.

[36] T. Tian and K. Burrage. Stochastic models for regulatory networks of the genetic toggle switch. *Proc. Natl. Acad. Sci. USA*, 103(22):8372–8377, 2006.

[37] R. B. Sidje. Expokit: A software package for computing matrix exponentials. *ACM Trans. Math. Softw.*, 24(1):130–156, 1998.

[38] B. Munsky, Z. Fox, and G. Neuert. Integrating single-molecule experiments and discrete stochastic models to understand heterogeneous gene transcription dynamics. *Methods*, 85:12–21, 2015.

[39] G. B. Leenders and J. A. Tuszynski. Stochastic and deterministic models of cellular p53 regulation. *Front. Oncol.*, 3:64, 2013.

[40] S. Ramaswamy, R. Lakerveld, P. I. Barton, and G. Stephanopoulos. Controlled formation of nanostructures with desired geometries: Part 3. dynamic modeling and simulation of directed self-assembly of nanoparticles through adaptive finite state projection. *Ind. Eng. Chem. Res.*, 54(16):4371–4384, 2015.

[41] K. N. Dinh and R. B. Sidje. Analysis of inexact krylov subspace methods for approximating the matrix exponential. *Mathematics and Computers in Simulation*, 138:1–13, 2017.

[42] A. Bouras and V. Fraysse. Inexact matrix-vector products in Krylov methods for solving linear systems: A relaxation strategy. *SIAM J. Matrix Anal. Appl.*, 26(3):660–678, 2005.

[43] J.V.D. Eshof and G.L.G. Sleijpen. Inexact Krylov subspace methods for linear systems. *SIAM J. Matrix Anal. Appl.*, 26(1):125–153, 2004.

[44] V. Simoncini and D.B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25(2):454–477, 2003.

[45] R.B. Sidje and N. Winkles. Evaluation of the performance of inexact GMRES. *J. Comp. Appl. Math.*, 235:1956–1975, 2011.

[46] E. Gallopoulos and Y. Saad. Efficient solution of parabolic equations by Krylov approximation methods. *SIAM J. Sci. Stat. Comput.*, 13(5):1236–1264, 1992.

[47] Y. Saad. Analysis of some Krylov subspace appproximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 29(1):209–228, 1992.

[48] M. Hochbruck and C. Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 34:1911–1925, 1997.

[49] M.A. Botchev, V. Grimm, and M. Hochbruck. Residual, restarting and Richardson iteration for the matrix exponential. *SIAM J. Sci. Comput.*, 35(3):A1376–A1397, 2013.

[50] L. Giraud, S. Gratton, and J. Langou. Convergence in backward error of relaxed GMRES. *SIAM J. Sci. Comput.*, 29(2):710–728, 2007.

[51] N.J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, Philadelphia, PA, USA, 2008.

[52] A.H. Al-Mohy and N.J. Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM J. Sci. Comput.*, 33(2):488–511, 2011.

[53] R.B. Sidje. Inexact uniformization and GMRES methods for large Markov chains. *Num. Lin. Alg. Appl.*, 18:947–960, 2011.

[54] J. Goutsias. Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems. *J. Chem. Phys.*, 122(18):184102, May 2005.

[55] E. Stirk, C. Molina-Paris, and H. van den Berg. Stochastic niche structure and diversity maintenance in the T cell repertoire. *Journal of theoretical biology*, 255:237–249, 2008.

[56] T. Vo and C. Priami. Simulation of biochemical reactions with time-dependent rates by the rejection-based algorithm. *The journal of chemical physics*, 143(054104), 2015.

[57] R. M. Anderson. *Population dynamics of infectious diseases: theory and applications*. Chapman and Hall, London-New York, 1982.

[58] D. J. Daley and J. Gani. *Epidemic modeling: an introduction*. Cambridge University Press, 2005.

[59] N. Bacaër. On the stochastic SIS epidemic model in a periodic environment. *Journal of mathematical biology*, 71(2):491–511, 2015.

[60] P. Bader, S. Blanes, F. Casas, and E. Ponsoda. Efficient numerical integration of Nth-order non-autonomous linear differential equations. *Journal of computational and applied mathematics*, 291:380–390, 2016.

[61] A. Leier and T. T. Marquez-Lago. Delay chemical master equation: direct and closed-form solutions. *Proceedings of the Royal Society of London A*, 471(20150049), 2015.

[62] K. N. Dinh and R. B. Sidje. An adaptive magnus expansion method for solving the chemical master equation with time-dependent propensities. *Journal of Coupled Systems and Multiscale Dynamics*, 2018.

[63] K. N. Dinh and R. B. Sidje. A comparison of the Magnus expansion and other solvers for the chemical master equation with variable rates. *AMMCS 2017 Conference proceedings*, 2018.

[64] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of physical chemistry*, 81:2340–2361, 1977.

[65] R. Purtan and A. Udrea. A modified Stochastic Simulation Algorithm for time-dependent intensity rates. In *19th International conference on control systems and computer science*, 2013.

[66] Y. Niu, K. Burrage, and L. Chen. Modelling biochemical reaction systems by stochastic differential equations with reflection. *Journal of theoretical biology*, 396:90–104, 2016.

[67] K. Burrage, P. Burrage, S. Leier, and T. Marquez-Lago. *Stochastic processes, multiscale modeling, and numerical methods for computational cellular biology*, chapter A review of stochastic and delay simulation approaches in both time and space in computational cell biology. Springer, 2017.

[68] W. Magnus. On the exponential solution of differential equations for a linear operator. *Communications on pure and applied mathematics*, 7(4):649–673, 1954.

[69] A. Iserles, S. P. Nørsett, and A. F. Rasmussen. Time symmetry and high-order Magnus methods. *Applied numerical mathematics*, 39(3-4):379–401, 2001.

[70] S. Blanes, F. Casas, J. A. Oteo, and J. Ros. The Magnus expansion and some of its applications. *Physics reports*, 470:151–238, 2009.

[71] A. Iserles, A. Marthinsen, and S. P. Nørsett. On the implementation of the method of Magnus series for linear differential equations. *BIT Numerical mathematics*, 39(2):281–304, 1999.

[72] N. Aparicio, S. Malham, and M. Oliver. Numerical evaluation of the evans function by Magnus integration. *BIT Numerical mathematics*, 45(2):219–258, 2005.

[73] K. Burrage. *Parallel and sequential methods for ordinary differential equations*. Clarendon Press, 1995.

[74] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations I, nonstiff problems*. Springer, 1987.

[75] G. Wanner and E. Hairer. *Solving ordinary differential equations II, stiff and differential algebraic problems*. Springer, 1991.

[76] G. Hall and A. Usman. Modified order and stepsize strategies in Adams codes. *Journal of computational and applied mathematics*, 11:113–122, 1999.

[77] L. F. Shampine and M. K. Gordon. *Computer solution of ordinary differential equations: the initial value problem*. W.H. Freeman and Co., 1975.

[78] R. W. Brankin, I. Gladwell, and L. F. Shampine. RKSUITE: a suite of Runge–Kutta codes for the initial value problem for ODEs, Softreport 91-1. Technical report, Math. Dept., Southern Methodist University, Dallas, TX, USA, 1991.

[79] P. N. Brown, G. D. Byrne, and A. C. Hindmarsh. VODE: A variable-coefficient ODE solver. *SIAM journal on scientific and statistical computing*, 10(5):1038–1051, 1989.

[80] Y. Cai, X. Peng, Q. Li, and K. Wang. A numerical solution to the nonlinear point kinetics equations using Magnus expansion. *Annals of nuclear energy*, 89:84–89, 2016.

[81] S. MacNamara and K. Burrage. Stochastic modeling of naive T cell homeostasis for competing clonotypes via the master equation. *Multiscale Modeling & Simulation*, 8(4):1325–1347, 2010.

[82] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):1–46, 2003.

[83] H. Vo and R. B. Sidje. Approximating the large sparse matrix exponential using incomplete orthogonalization and Krylov subspaces of variable dimension. *Numerical linear algebra with applications*, 24(3):e2090, 2017.

[84] R. Cools. Constructing cubature formulae: the science behind the art. *Acta Numerica*, 6:1–54, 1997.

[85] P. Davis and P. Rabinowitz. *Methods of numerical integration*. Dover publications, Inc., 1984.

[86] R.B. Sidje and W.J. Stewart. A numerical study of large sparse matrix exponentials arising in Markov chains. *Comput. Stat. Data Anal.*, 29:345–368, 1999.

[87] K. N. Dinh and R. B. Sidje. A comparison of the Magnus expansion and other solvers for the chemical master equation with time-dependent propensities. (in preparation), 2017.

[88] R. Johnson and B. Munsky. The finite state projection approach to analyze dynamics of heterogeneous populations. *Physical Biology*, 14(3), 2017.

[89] K. N. Dinh and R. B. Sidje. An application of the krylov-fsp-ssa method to parameter fitting with maximum likelihood. *Physical Biology*, 14(065001), 2017.

[90] T. Tian, S. Xu, J. Gao, and K. Burrage. Simulated maximum likelihood method for estimating kinetic rates in gene expression. *Bioinformatics*, 23(1):84–91, 2007.

[91] S. Poovathingal and R. Gunawan. Global parameter estimation methods for stochastic biochemical systems. *BMC bioinformatics*, 11:414, 2010.

[92] B. Daigle Jr., M. Roh, L. Petzold, and J. Niemi. Accelerated maximum likelihood parameter estimation for stochastic biochemical systems. *BMC bioinformatics*, 13(68), 2012.

[93] Y. Wang, S. Christley, E. Mjolsness, and X. Xie. Parameter inference for discretely observed stochastic kinetic models using stochastic gradient descent. *BMC Systems biology*, 4(99), 2010.

[94] A. Horvath and D. Manini. Parameter estimation of kinetic rates in stochastic reaction networks by the em method. In *Proceedings of the 2008 International conference on Biomedical engineering and informatics*, volume 1, pages 713–717, 2008.

[95] S. Reinker, R. Altman, and J. Timmer. Parameter estimation in stochastic biochemical reactions. *Systems biology*, 153(4):168–178, 2006.

[96] Z. Fox, G. Neuert, and B. Munsky. Finite state projection based bounds to compare chemical master equation models using single-cell data. *Journal of chemical physics*, 145(7):074101, 2016.

[97] T. Ellis, X. Wang, and J. J. Collins. Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nature biotechnology*, 27:465–471, 2009.

[98] H. McAdams and A. Arkin. It's a noisy business! genetic regulation at the nanomolar scale. *Trends in genetics*, 15(2):65–69, 1999.

[99] D. Fange and J. Elf. Noise-induced min phenotypes in e. coli. *PLoS Computational biology*, 2(e80), 2006.

[100] M. Samoilov, S. Plyasunov, and A. Arkin. Stochastic amplification and signaling in enzymatic futile cycles through noise-induced bistability with oscillations. *PNAS*, 102(7):2310–2315, 2005.

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

[102] A. Colman-Lerner, A. Gordon, E. Serra, T. Chin, O. Resnekov, D. Endy, C. Pesce, and R. Brent. Regulated cell-to-cell variation in a cell-fate decision system. *Nature*, 437:699–706, 2005.

[103] I. Golding, J. Paulsson, S. Zawilski, and E. Cox. Real-time kinetics of gene activity in individual bacteria. *Cell*, 123(6):1025–1036, 2005.

[104] J. Yu, J. Xiao, X. Ren, K. Lao, and X. Lie. Probing gene expression in live cells, one protein molecule at a time. *Science*, 311(5767):1600–1603, 2006.

[105] T. Gardner, C. Cantor, and J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403:339–342, 2000.

[106] E. Yang, E. van Nimwegen, M. Zavolan, N. Rajewsky, M. Schroeder, M. Magnasco, and J. Darnell Jr. Decay rates of human mrnas: correlation with functional characteristics and sequence attributes. *Genome research*, 13(8):1863–1872, 2003.

[107] I. Chou and E. Voit. Recent developments in parameter estimation and structure identification of biochemical and genomic systems. *Mathematical Biosciences*, 219(2):57–83, 2009.

[108] D. Wilkinson. Stochastic modelling for quantitative description of heterogeneous biological systems. *Nature*, 10:122–133, 2009.

[109] B. Munsky and M. Khammash. Identification from stochastic cell-to-cell variation: a genetic switch case study. *IET systems biology*, 4(6):356–366, 2010.

[110] H. Xu, S. Skinner, A. Sokac, and I. Golding. Stochastic kinetics of nascent rna. *Physical review letters*, 117(128101), 2016.

[111] R. Boys, D. Wilkinson, and T. Kirkwood. Bayesian inference for a discretely observed stochastic kinetic model. *Statistics and computing*, 18(2):125–135, 2008.

[112] A. Golightly and D. Wilkinson. Bayesian sequential inference for stochastic kinetic biochemical network models. *Journal of computational biology*, 13(3):838–851, 2006.

[113] L. Rios and N. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56:1247–1293, 2013.

[114] C. Moles, P. Mendes, and J. Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome research*, 13:2467–2474, 2003.

[115] R. Brent. *Algorithms for finding zeros and extrema of functions without calculating derivatives*. PhD thesis, Department of Computer science, Stanford University, 1971.

[116] J. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[117] R. O'Neill. Algorithm as 47: function minimization using a simplex procedure. *Journal of the Royal statistical society - series C (Applied statistics)*, 20(3):338–345, 1971.

[118] M. J. D. Powell. The newuoa software for unconstrained optimization without derivatives. Technical Report NA05, Department of Applied Mathematics and Theoretical Physics, Cambridge University, 2004.

[119] M. J. D. Powell. Least frobenius norm updating of quadratic models that satisfy interpolation conditions. *Mathematical programming*, 100:183–215, 2004.

[120] C. Boender, A. Rinnooy Kan, G. Timmer, and L. Stougie. A stochastic method for global optimization. *Mathematical programming*, 22(1):125–140, 1982.

[121] G. Timmer. *Global optimization: a stochastic approach*. PhD thesis, Erasmus University Rotterdam, 1984.

[122] T. Csendes. Nonlinear parameter estimation by global optimization - efficiency and reliability. *Acta Cybernetica*, 8(4):361–370, 1988.

[123] W. Goffe, G. Ferrier, and J. Rogers. Global optimization of statistical functions with simulated annealing. *Journal of econometrics*, 60:65–99, 1994.

[124] W. Goffe. Simann: a global optimization algorithm using simulated annealing. *Studies in nonlinear dynamics and econometrics*, 1(3):169–176, 2007.

[125] M. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of physical chemistry A*, 104(9):1876–1889, 2000.

[126] Y. Cao, H. Li, and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *Journal of chemical physics*, 121(9):4059, 2004.

[127] J. McCollum, G. Peterson, C. Cox, M. Simpson, and N. 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.

[128] H. D. Vo and R. B. Sidje. An adaptive solution to the chemical master equation using tensors. *Journal of chemical physics*, 147:044192, 2017.

[129] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, and R. Marti. Scatter search and local nlp solvers: a multistart framework for global optimization. *INFORMS Journal on computing*, 19(3):328–340, 2007.

[130] F. Glover. *Artificial evolution*, volume 1363, chapter A template for scatter search and path relinking, pages 13–54. Springer, 1998.

[131] A. Khachaturyan, S. Semenovsovskaya, and B. Vainshtein. The thermodynamic approach to the structure analysis of crystals. *Acta Crystallographica*, A37:742–754, 1981.

[132] S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[133] V. Cerny. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[134] S. Semenovskaya, K. Khachaturyan, and A. Khachaturyan. Statistical mechanics approach to the structure determination of a crystal. *Acta Crystallographica*, A41:268–273, 1985.