Analysis and Numerics of the Chemical Master Equation

Vikram Sunkara

03 2013

A thesis submitted for the degree of Doctor of Philosophy of the Australian National University
For my ever loving and supportive family; my father, my mother and my older brother.
Declaration

The work in this thesis is my own except where otherwise stated.

Vikram Sunkara
Acknowledgements

First and foremost I would like to acknowledge my supervisor Professor Markus Hegland. Professor Hegland has provided me with great postgraduate training and support throughout my three years of study. Through the roller-coaster ride of writing my thesis, he has been a great support and guide, aiding me through my difficulties. He has given me great support in travelling overseas to conduct research in groups giving me first hand experience of international research. Most of all I would like to acknowledge Professor Hegland for always supporting me in exploring new ideas and for helping me transform my graphite like ideas into something more precious.

I acknowledge Dr Shev MacNamara and Professor Kevin Burrage for their inspiration and discussions in the start of my Ph.D. I got a great insight into the CME in my research visits to UQ and in my visit to Oxford.

I acknowledge Rubin Costin Fletcher who had helped me tremendously by initiating the implementation of CMEPy and giving me great assistance in learning and adapting CMEPy to empirically prove my methods. Discussions with him during his stay at ANU were most valuable.

I acknowledge Dr Paul Leopardi, who helped me voice out my ideas and proofs and also for being a close friend in time of great personal distress.

I acknowledge Jahne Meyer and Abhinay Mukunthan, who through the three years, have helped me tremendously in proof reading the papers that I have written. Their feedback was most valuable and the cartoons drawn by Mr Mukunthan were ever so essential.

I acknowledge the IT team at the ANU mathematics department, Nick Gouth and Shi Bai. Computing infrastructure was crucial in my work and Nick and Shi Bai aided me in always having the resources that I needed to conduct the research I needed. Their continuous support is ever so missed when at any other institute.

Finally I would like to thank my lab mates over the three years, who through many discussions and pleasant conversations were always there to support me.
Shi Bai, Srinivasa Subramanyam, ChinFoon Koo, Sudi Mungkasi and Chris.
Abstract

It is well known that many realistic mathematical models of biological and chemical systems, such as enzyme cascades and gene regulatory networks, need to include stochasticity. These systems can be described as Markov processes and are modelled using the Chemical Master Equation (CME). The CME is a differential-difference equation (continuous in time and discrete in the state space) for the probability of a certain state at a given time. The state space is the population count of species in the system. A successful method for computing the CME is the Finite State Projection Method (FSP). The purpose of this literature is to provide methods to help enhance the computation speed of the CME. We introduce an extension to the FSP method called the Optimal Finite State Projection method (OFSP). The OFSP method keeps the support of the approximation close to the smallest theoretical size, which in turn reduces the computation complexity and increases speed-up. We then introduce the Parallel Finite State Projection method (PFSP), a method to distribute the computation of the CME over multiple cores, to allow the computation of systems with a large CME support. Finally, a method for estimating the support a priori is introduced, called the Gated One Reaction Domain Expansion (GORDE). GORDE is the first domain selection method in the CME literature which can guarantee that the support proposed by the method will give the desired FSP approximation error.

To prove the accuracy and convergence of these three methods, we explore non-linear approximation theory and the theory of Chemical Master Equations via reaction counts. Using these tools, the proofs of the accuracy and convergence of the three methods are given. Some numerical implementations of the three methods are given to demonstrate experimental speed-up in computing an approximation of the CME.
# Contents

<table>
<thead>
<tr>
<th>Acknowledgements</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ix</td>
</tr>
<tr>
<td>Notation and terminology</td>
<td>xiii</td>
</tr>
</tbody>
</table>

## 1 Introduction 1

## 2 Chemical Master Equation 7

2.1 The Chemical Master Equation 7

2.1.1 The CME Problem 13

2.1.2 Biological Examples 14

2.2 CME Structure 17

2.3 Finite State Projection Method 20

2.3.1 Sink State 22

2.3.2 $N$-Step algorithm 25

2.3.3 Sliding Window Algorithm 25

2.4 Stochastic Simulations 27

2.5 Aggregation 32

2.6 The $\ell_2$ Projections of the CME 34

## 3 Optimal Finite State Projection Method 39

3.1 Best $N$-term approximation 40

3.1.1 $\mathcal{A}^s$ Spaces, $s > 0.$ 41

3.1.2 Weak $\ell_r$ spaces, $0 < r < 1.$ 42

3.1.3 Best $N$-term approximation bound 44

3.2 Optimal Finite State Projection method 47

3.3 Numerical Experiments 49

3.3.1 Goutsias’ Model 50
Notation and terminology

The following are the fixed notations in the text.

### Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_s$</td>
<td>Number of distinct species</td>
</tr>
<tr>
<td>$N_r$</td>
<td>Number of distinct species</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>State Space, subset of $\mathbb{N}_0^{N_s}$</td>
</tr>
<tr>
<td>$x$</td>
<td>state in the state space, $x \in \Omega$</td>
</tr>
<tr>
<td>$P(x; t)$</td>
<td>Probability of the system being at state $x$ at time $t$.</td>
</tr>
<tr>
<td>$R_k$</td>
<td>mapping of a state to the next state under the $k$th reaction.</td>
</tr>
<tr>
<td>$\alpha_k$</td>
<td>non-negative propensity functions</td>
</tr>
<tr>
<td>$v_k$</td>
<td>Stochiometric transition vector of reaction $R_k$.</td>
</tr>
<tr>
<td>$\mathbb{V}$</td>
<td>Stochiometric matrix</td>
</tr>
<tr>
<td>$A$</td>
<td>Infitesimal Generator</td>
</tr>
<tr>
<td>$p_t$</td>
<td>Probability distribution over the state space $\Omega$.</td>
</tr>
<tr>
<td>$A'$</td>
<td>is a finite submatrix of $A$.</td>
</tr>
<tr>
<td>$\ell_2(\Omega)$</td>
<td>Hilbert space defined over finite state space $\Omega$.</td>
</tr>
<tr>
<td>$\nabla$</td>
<td>Discrete domain.</td>
</tr>
<tr>
<td>$\sum_N$</td>
<td>nonlinear subspace of $\ell_2(\nabla)$</td>
</tr>
<tr>
<td>$A_s$</td>
<td>Approximation space</td>
</tr>
</tbody>
</table>
supp  Support of a vector, set of all non zero valued indexes.

$v^*$  Decreasing rearrangement of a vector $v \in \ell_2$

$\ell_{r,\infty}$  Weak $\ell_r$ space

$Z$  State space partitioning map.

$\Lambda$  Reactions state space, subset of $\mathbb{N}_0^N$

$r$  Reaction state, $r \in \Lambda$

$\Gamma_x$  Affine mapping from species to the reaction state space with starting point $x \in \Omega$

$t$  Time

$\Delta t$  Time step
## Terminology

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CME</td>
<td>Chemical Master Equation</td>
</tr>
<tr>
<td>FSP</td>
<td>Finite State Projection method</td>
</tr>
<tr>
<td>SSA</td>
<td>Stochastic Simulation Algorithm</td>
</tr>
<tr>
<td>OFSP</td>
<td>Optimal Finite State Projection method</td>
</tr>
<tr>
<td>SW</td>
<td>Sliding Window Algorithm</td>
</tr>
<tr>
<td>PFSP</td>
<td>Parallel Finite State Projection method</td>
</tr>
<tr>
<td>GORDE</td>
<td>Gated One Reaction Domain Expansion</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Deterministic models do not provide an accurate description of biological systems with a small number of interacting species [55, 85, 50, 76]. The addition of stochasticity provides an accurate description of these systems. The introduction of stochasticity in models has made significant contributions to the fields of neuroscience [69], T-cell homeostasis [61, 20, 78], signalling of cells [73, 50], viral invasion of cells [68, 87, 76] and in-silico modelling of drug effectiveness [66, 5].

In a typical biological system, there exist numerous different species which undergo changes in population due to stochastic reactions firing. In a mixture of species, different particles diffuse randomly and interact with other species which they come in contact with. The occurrence of reactions follow a Poisson process. When the particle numbers are high, the stochasticity can be ignored and the system can be modelled deterministically. However, when the particle numbers are small, stochasticity plays an important role.

The focus of this thesis is on the class of biological systems whose population can be modelled over time as a jump Markov process, (c.f. Definition 2.2). The state space of the jump Markov process is considered to be all the different population configurations that the system can undertake. Furthermore, the population in the biological system changes due to reactions which fire following a Poisson process.

Since the reactions fire stochastically, there does not exist a single solution which accurately predicts the evolution of the population. There are a class of events which could occur, with each sequence of firing reactions leading to a different realisation of the biological system. A probability measure is then required to classify each realisation according to its probability with respect to the others. The solution of the Chemical Master Equation (CME) is a probability
distribution over the entire state space, describing the probability of the biological system realising a particular state at a particular point in time.

The CME is a difference differential equation. Let us denote $x$ as the tuple with entries as the populations of each species, let $t > 0$, then the CME at state, $x$, for time $t > 0$ is given by

$$\frac{\partial P(x; t)}{\partial t} = \sum_{k=1}^{N_r} \alpha_k(x - v_k)P(x - v_k; t) - \sum_{k=1}^{N_r} \alpha_k(x)P(x; t).$$

The parameters for this equation are given in more detail in §2.1.1. Considering over all the possible population configurations $x$, we get the initial value problem

$$\begin{align*}
\frac{dp_t}{dt} &= Ap_t, \quad t > 0, \\
p_0 := (\delta_{x,x_0})_{x \in \Omega}, \quad t = 0.
\end{align*}$$

(1.1)

We discuss this further in §2.1.1. The solution to this problem is the matrix exponential $e^{At}$, however, the computation of this operator is dubious. There are thus two methods of solving the CME, normalising over numerous stochastic simulations and approximating the operator $e^{At}$. We discuss the different methods more extensively in chapter 2.1.

The CME is computed for two particular applications, in-silico simulation of biological systems and parameter fitting experimental data.

The current research focus is on the development of methods to fit stochastic models to the experimental data and estimate the unknown parameters of the experiments [51, 64]. For parameter estimation with methods such as Maximum Likelihood and Expectation Maximisation, the probability distribution over the state space is needed. Hence, computing the solution of the Chemical Master Equation (CME) is an important problem.

The Chemical Master Equation is computationally difficult. However, between 2005 and 2009, key methods were introduced to help simplify the computation of the CME. The methods are:

- The Finite State Projection method, in which the CME is only solved over a truncated state space;
- Aggregation methods, where partitions of the state space are lumped together to generate a smaller problem;
- Stochastic Simulations, where multiple realisations are taken and an empirical distribution is generated; and
• $\ell_2$ methods, where the CME is reformulated in $\ell_2$ and solutions are found in terms of basis coefficients.

These methods compute approximations of the CME for systems up to 10 dimensions in the order of tens of minutes [46, 42, 59, 60, 31]. Since the domain describes different combinations of species population, the order of the state space is dimensionally exponential. For example, the state space of a system with 10 different types of particles, where each particle can be alive or dead, is the size of $2^{10}$. The state space for biological systems with more than 10 varieties of species becomes too large to compute. As larger dimensional problems are considered, it becomes clear that adaptability and subdivision of the problem are essential tools in constructing approximations of the CME. Adaptability is required to constructively find the smallest problem to compute and subdivision of the problem is required to construct tractable sub-problems, which can then be distributed over a high performance computing (HPC) cluster.

The following questions were considered open problems in 2009 and it was necessary to address these questions in order to construct an adaptive framework for the CME and expand numerical approximations to higher dimensions.

Q1) What is the smallest subset of the state space on which to construct an approximation of the CME?

Q2) How do we parallelise the CME to compute larger problems?

Q3) Should the CME be solved by constructing an empirical distribution from stochastic simulations or by approximating the matrix exponential?

The solution for Question 1 enables the computation of the smallest support for a given error tolerance. It was experimentally shown that the smallest support grows polynomial in time for certain problems [81]. However, in practice, a hyper-rectangular support is constructed by default if the support is not adaptively chosen. The use of a hyper-rectangular support results in an exponential growth of the support at higher dimensions. This in turn makes simple problems unfeasible to compute.

Question 2 addresses the issues arising when we want to solve the CME for large dimensions ($\gg 10$). In these scenarios, the minimal domain is itself in the order of millions or billions. Hence, it is important to devise a method of sub-dividing the problem into smaller problems and compute each problem independently.
Question 3 is a running debate in the CME community. Approximation of the CME via stochastic simulations is a simple and popular methodology. This is because the two hardest aspects of solving via approximating the matrix exponential are trivial when solving via stochastic simulation. These two aspects are adaptivity and sub-division. Since the stochastic simulation constructs the approximation via realisations, each realisation realises the support as it progresses, thus wasting no time in search of the right domain. Each realisation is also independent of the other, allowing realisations to be run in parallel over as many cores as possible. Furthermore, the independence between realisations guarantees a linear speed up in computation time when using multi-core architectures. While the stochastic simulation methods are easier to implement, for dimensions less than 30, it is shown for some problems that solving via approximation of the matrix exponential is quicker than solving via stochastic simulation.

The questions thus arise, “What are the conditions under which the stochastic simulation will be faster than approximating matrix exponentials?”, specifically, “Is there some dimension after which one should only consider stochastic simulations?”

Finding solutions to the three questions is essential in accelerating the existing numerical methods that are presented in the literature and also help increase the class of biological systems for which the CME can be approximated.

The three questions are answered through this thesis, however the answers of three question are logically connected as follows:

It is shown for the first time in Theorem 4.1 that the number of realisations needed is always less than or equal to the minimal support for approximations of the same error. Hence, even for a system with a large number of dimensions, the approximation via matrix exponential is a smaller problem than approximation via stochastic simulations. This is dependent on the knowledge of the minimal support. Thus, the answer to question three is; the matrix exponential is the method of solving the CME if we know what its similar support is. The solution to question 3 is therefore linked to Question 1.

To answer question 1, we require a method of determining the minimal support of an approximation of the CME. In 2010, Sunkara and Hegland proposed such a method called the Optimal Finite State Projection method (OFSP). They showed that the support of the OFSP method’s approximation of optimal order, that is, the approximation’s support is a constant away from the minimal support. The OFSP method guarantees optimal order support of the approxima-
tion by first constructing a hyper-rectangle and then truncating it. The OFSP method can compute small dimensional problems a few orders of magnitude faster than conventional FSP. However, a tighter domain selection method is needed for higher dimensional problems. Hence the Gated One Reaction Domain Expander (GORDE), was proposed to compute the support for a future time step. The GORDE algorithm is the support selection algorithm for the CME which guarantees a priori that the proposed support will give the desired FSP approximation error. While other domain selection methods have been proven to converge, they cannot guarantee a priori that the proposed domain produces the desired FSP error. By combining the GORDE and the OFSP algorithm, we thus construct an optimal order support, answering question 1.

When approximating the CME for these systems, the stochastic simulation can be easily parallelised. The GORDE algorithm and the OFSP give the minimal support. However, if the minimal support is of the order of million or billions, we need to parallelise it to keep the matrix exponential approximation competitive with the stochastic simulation methods. Hence, in 2010, Sunkara and Hegland introduced the framework of the Parallel Finite State Projection method (PFSP). Sunkara and Hegland used the linearity of the matrix exponential to subdivide the problem and construct approximations of the CME on multiple cores in parallel.

The previous paragraphs described the GORDE algorithm to accurately estimate the support, the OFSP method to construct an approximations with optimal order support and the PFSP method to compute a new class of biological problems. Combining these three methods, we have a new framework for solving the CME problem. This new framework can be applied to QSSA, Krylov FSP, Aggregation and Wavelet methods for enhancing and speeding up the approximation.

The rest of this thesis is organised as follows. We formulate the Chemical Master Equation and then describe some of the key frameworks for solving it in Chapter 2.1. We then derive the Optimal Finite State Projection method (OFSP) and prove the optimal order result in Chapter 3. In Chapter 4, we derive the Parallel Finite State Projection method (PFSP) and prove that the sub-problems behave nicely and provide the required error bounds. Lastly, we present the Gated One Reaction Domain Expander (GORDE) and introduce a formulation of the CME with respect to reaction counts to help prove the accuracy and convergence of the GORDE algorithm.
CHAPTER 1. INTRODUCTION
Chapter 2

Chemical Master Equation

2.1 The Chemical Master Equation

Deterministic descriptions of interacting particle, in small population numbers, was shown to be inadequate\cite{56, 85, 50, 76}. When small number of interactions are occurring, it was shown that a stochasticity play a crucial role in describing these interactions. In the biological and chemical systems with small number of species/particle interactions, we assume that the reactions which are firing to alter the system are firing stochastically. We define a stochastic process below.

Definition 2.1. A probability space is defined as a triple \((S, F, P)\). In the triple, \(S\) denotes a sample space, \(F\) denotes a \(\sigma\)-algebra made up of subsets of \(S\) and \(P : F \rightarrow [0, 1]\) is the probability measure. \(\Omega\).

Definition 2.2. Given a probability space \((S, F, P)\), a stochastic process \(X\) is a family of random variables \(\{X(t), t \in \mathcal{T}\}\). Here each random variables \(X(t)\) is defined over \(\Omega \subset S\). We call \(\Omega\), the values of the sample space the random variable \(X(t)\) can be, the state space. Here \(\mathcal{T}\) is the (time) index set of the process. For a given \(t \in \mathcal{T}\) and \(x \in \Omega\), we denote \(P(X_t = x)\) as the probability of the stochastic process being at state \(x\) at time \(t\).

Definition 2.3. Let \(\{X(t), t \in \mathcal{T}\}\), be a stochastic process with probability space \((S, F, P)\). Let \(t_0 < t_1 < \ldots < t\) be \(n + 1\) time points in \(\mathcal{T}\). Let \(x_0, \ldots, x_{n-1}, x \in \Omega\), with the indices corresponding to the time point indices. Then a Stochastic process \(\{X(t), t \in \mathcal{T}\}\), is said to be a Markov process if for all \(t \in \mathcal{T}\) and \(x \in \Omega\),

\[
P(X(t) = x|X(t_{n-1}) = x_{n-1}, X(t_{n-2}) = x_{n-2}, \ldots, X(t_0) = x_0) = P(X(t) = x|X(t_{n-1}) = x_{n-1}).
\]
Definition 2.4. A stochastic process \( \{X(t), t \in \mathcal{T}\} \), with probability space \((S, F, P)\), is said to be a jump process if the \( \Omega \) is a discrete set. That is the process takes discrete jumps between states in the state space over time.

Our focus is only on continuous time stochastic processes, that is \( \mathcal{T} = [0, \infty) \). From here all stochastic processes are implicitly continuous time stochastic processes. Also for simplicity denote a stochastic process as \( X(t) \) rather than \( \{X(t), t \in \mathcal{T}\} \), as our index set is fixed to \([0, \infty)\).

The biological and chemical problems we are interested in studying have the characteristic that the reactions driving the system are given by a counting process, that is, a stochastic process whose state space is the non-negative integers and the process is non-decreasing in time. Since the reactions are firing according to a counting process, then the population of the species in the system is given by jump Markov process. A jump Markov process is a jump process which is also a Markov process. The biological and chemical systems we are interested in are systems with their population configuration described by a jump Markov process.

The Chemical Master Equation (CME) solves for the probability distribution over the state space of a jump Markov process. The CME is derived by substituting the transitional probability of the jump Markov process into the Chapman-Kolmogorov equation. The CME has applications such as ion channel gating, RNA transduction, T-cell homeostasis are biological systems where the particle states are changing and the changes are being driven stochastically via independent reactions/interactions.

For demonstration, let us consider a simple system where a particle can undergo birth and decay. We begin with the ODE description of this system, then use the Kurtz formulation \([7]\) to derive the CME of that system. Note that the birth and decay process of a single particle is only one dimensional in population. The following steps is ansatz for higher dimensional systems.

Let \( Z(t) \) be the population of a particle \( Z \) at time \( t > 0 \), and let the population be governed by,

\[
\frac{dZ(t)}{dt} = -\alpha Z(t) + \beta,
\]

where \( \alpha, \beta > 0 \). The equation above is interpreted as a change in \( Z \), caused by two reactions. The first reaction being a decaying process, which removes particle \( Z \) at rate \( -\alpha Z(t) \). The other reaction is a birth reaction which introduce \( Z \) at rate \( \beta \). The terms \( \alpha Z(t) \) and \( \beta \) are called reaction rates. We then have
2.1. THE CHEMICAL MASTER EQUATION

\[
\frac{dZ(t)}{dt} = -\alpha Z(t) + \beta,
\]

\[
Z(t) = \hat{Z}(0) - \int_0^t \alpha Z(s) ds + \int_0^t \beta ds.
\]

Kurtz et al propose that the contributions of the reactions in the equation above are replaced with two counting processes, \( R_1 \) and \( R_2 \). Hence we approximate \( Z(t) \) by a jump Markov process, denoted by \( X(t) \) and defined as

\[
X(t) := X(0) - R_1(t) + R_2(t),
\]

(2.2)

where \( X(0) = Z(0) \) and \( R_1(t), R_2(t) \) are given by,

\[
R_1(t) := Y_1 \left( \int_0^t \alpha X(s) ds \right),
\]

(2.3)

\[
R_2(t) := Y_2 \left( \int_0^t \beta ds \right),
\]

(2.4)

where \( Y_1 \) and \( Y_2 \) are independent unit Poisson processes.

The approximation \( X(t) \) states that the change in populations of \( Z(t) \) is minus the number of decays that have occurred and plus the number of births that have occurred. The number of births and decays are given by a counting processes described in (2.3) and (2.4). The process \( X(t) \) is called the first scale stochastic approximation of \( Z(t) \), (2.1), [7, 55].

Now we derive the probability distribution of the stochastic process \( X(t) \). Let \( T_{\Delta t}(x_2|x_1) \) denote the transition probability of a stochastic process, that is, the probability of the stochastic process transitioning from state \( x_1 \) to \( x_2 \) in time \( \Delta t > 0 \). The process \( X \) is the sum of independent unit Poisson processes. The transition probability of \( X \) from \( x_1 \) to \( x_2 \), with \( x_1, x_2 \in \mathbb{N}_0 \), and \( \Delta t > 0 \), is

\[
T_{\Delta t}(x_2|x_1) = \delta_{x_1+1,x_2} \beta \Delta t + \delta_{x_1,x_2} (1 - (\alpha x_1 + \beta) \Delta t) + O(\Delta t^2),
\]

(2.5)

where the \( \delta \) is a Kronecker delta. The derivation of the transition probability of counting processes is given in Appendix B. However, for the purpose of current description, we only need to know what the transition probability is stating. The transition probability of a stochastic process, \( X \), transitioning from \( x_1 \) to \( x_2 \) in a time interval \( \Delta t \) is the sum of the probability of the following three events occurring:
1. $X$ transitions from $x_1$ to $x_2$ via one reaction,

2. $X$ starting at $x_1$ and no reactions occurring in the interval $\Delta t$,

3. $X$ transitions from $x_1$ to $x_2$ via two or more reactions occurring.

The probabilities of these events are $\delta_{x_1+1,x_2} \beta \Delta t$, $\delta_{x_1,x_2} (1 - (\alpha x_1 + \beta) \Delta t)$ and $O(\Delta t^2)$ respectively. The derivation of the probabilities for the events listed above are given in Appendix B or widely found in standard literature by Cox and Miller [15].

We substitute (2.5) into an equation called the Chapman-Kolmogorov Equation to get the CME of the jump Markov process $X(t)$. The Chapman-Kolmogorov (C-K Equation) equation is given as the following; fix $x_i \in \Omega$ for $i \in \{1, 2, 3\}$ and let $X(0) = x_0$ be the starting state. For $t, \Delta t > 0$, the Chapman-Kolmogorov (C-K Equation) equation is,

$$T_{t+\Delta t}(x_2|x_0) = \sum_{x_1 \in \Omega} T_{\Delta t}(x_2|x_1) T_t(x_1|x_0).$$

(2.6)

By substituting $T_{\Delta t}(x_2|x_1)$ from (2.5) into the C-K Equation, $T_{t+\Delta t}(x_2|x_0)$ reduces to,

$$T_{t+\Delta t}(x_2|x_0) = \sum_{x_1 \in \Omega} (\delta_{x_1+1,x_2} \beta \Delta t) T_t(x_1|x_0)
+ \sum_{x_1 \in \Omega} (\delta_{x_1,x_2} (1 - (\alpha x_2 + \beta) \Delta t)) T_t(x_1|x_0)
+ \sum_{x_1 \in \Omega} (O(\Delta t^2)) T_t(x_1|x_0)
= \beta \Delta t T_t(x_2-1|x_0) + \alpha \Delta t (x_2+1) T_t(x_2+1|x_0) + (1 - (\alpha x_2 + \beta) \Delta t) T_t(x_2|x_0) + O(\Delta t^2).$$

(2.8)
2.1. THE CHEMICAL MASTER EQUATION

The starting state, $x_0$, is fixed at time $t = 0$ of $X(t)$, hence, $T_t(x_2, x_0)$ is written as $P(x_2; t)$ for simplicity. Then the equation (2.8) changes to,

$$
\frac{\partial P(x_2; t)}{\partial t} = \beta P(x_2 - 1; t) + \alpha (x_2 + 1) P(x_2 + 1; t) - (\alpha x_2 + \beta) P(x_2; t). \quad (2.9)
$$

The solution to (2.9) for $x \in \Omega (\Omega = \mathbb{N}_0)$ and $t > 0$ with initial population of $x_0 \in \mathbb{N}_0$ is,

$$
P(x; t) = \min \{x, x_0\} \sum_{k=0}^{\min \{x, x_0\}} \binom{x_0}{k} e^{-k\alpha t}(1 - e^{-\alpha t})^{x_0-k} \frac{\lambda(t)^{x-k}e^{\lambda(t)}}{(x-k)!}, \quad (2.10)
$$

with

$$
\lambda(t) = \frac{\beta(1 - e^{-\alpha t})}{\alpha}.
$$

The equation above gives the probability of the stochastic process $X(t)$ being $x \in \Omega$ at time $t > 0$, assuming that two driving reactions in the system are counting processes.

The steps above are ansatz to systems with multiple species. The steps are extended to high dimensional problems using the result by Kurtz [54]; that for a vector of counting processes, for each component of the vector there exists an independent unit processes such that the counting processes can be given by the unit Poisson processes.

There are many different stochastic biological/chemical systems, however we are interested in a particular class of system. That is we are interested in systems with discrete particle interactions, where the interactions are being driven by reactions which are firing in a Poisson fashion. Furthermore, the systems we are interested in need to have the following parameters:

Let a system have $N_s \in \mathbb{N}$ species and $N_r \in \mathbb{N}$ reactions. We define $X_{N_s}(t), \ t \in [0, \infty)$, to be a jump Markov process of $N_s$ dimension with state space $\Omega \subset \mathbb{N}_0^{N_s}$.

We define a reaction as an affine transform which changes the population configuration of the system. For each reaction indexed by $k = 1, \ldots, N_r$, we define the linear transformation of the $k$th reaction by $r_k : \Omega \to \Omega$.

For each reaction indexed by $k = 1, \ldots, N_r$, the deterministic rate at which the reaction fires is called the propensity. The propensity is a non-negative function and for indexes $k = 1, \ldots, N_r$ the propensity of each reaction $r_k$ is denoted by $\alpha_k : \mathbb{N}_0^{N_s} \to \mathbb{R}^+$, where

$$
\alpha_k(x) := \begin{cases} 
\geq 0 & \text{for } x \in \Omega, \\
0 & \text{otherwise.} 
\end{cases} \quad (2.11)
$$
For each reaction index $k = 1, \ldots, N_r$, $v_k$ is the stoichiometric vector of the $k$th reaction. That is $v_k$ is the difference in the population state, $x$, when reaction $k$ occurs, more specifically, $v_k := r_k(x) - x$, for all $x \in \Omega$. By writing the stoichiometric vectors as rows of a matrix gives us the stoichiometric matrix, which we denote by $V$.

In summary, the system has $N_s$ different particles which are undergoing $N_r$ reactions. The reactions are being driven by $N_r$ propensity functions and the changes in population by each reaction is given by the stoichiometric vectors $v_k$.

If we denote the population of the system at any point in time $t > 0$ as $X(t)$, then $X(t)$ must be given by the equation

$$X(t) = X(0) + \sum_{k=1}^{N_r} Y_k \left( \int_0^t \alpha_k(X(s)) \, ds \right) v_k,$$

where $X(0)$ is the initial population of the system and $Y_k$ for $k = 1, \ldots, N_r$ is a unit Poisson process [7]. The biological/chemical problems of importance for this thesis are ones which can be described by (2.12). To find the probability of such a system being in some state at a particular point in time, needs the substitution of the transition probability of $X(t)$ substituted into the Chapman-Kolmogorov equation. We show this below.

For a fixed $x_0, x_1 \in \Omega$, the transition probability of going from $x_0$ to $x_1$ in time $\Delta t > 0$ is given by,

$$T_{\Delta t}(x_1|x_0) = \sum_{k=1}^{N_r} \delta_{x_1,x_0} \alpha_k(x_0) \Delta t + \left( 1 - \sum_{k=1}^{N_r} \alpha_k(x_0) \Delta t \right) \delta_{x_1,x_0} + O(\Delta t^2).$$

The structure of the transition probability above is similar to the birth-decay example given above (2.5). The terms of (2.13) are as follows: for each reaction $k = 1, \ldots, N_r$, $\alpha_k(x_0) \Delta t$ is the probability of arriving at $x_1$ from $x_0$ via one reaction. The term, $1 - \sum_{k=1}^{N_r} \alpha_k(x_0) \Delta t$, is the probability of being at state $x_1$ and no reactions firing in time $\Delta t$. Lastly, the probability of reaching $x_1$ from $x_0$ by more than two reaction is $O(\Delta t^2)$. The derivation of transition probability is given in Appendix B.

Substituting the transition probability given above into the Chapman-Kolmogorov equation gives the chemical master equation (CME) of the stochastic processes $X(t)$,

$$\frac{\partial P(x; t)}{\partial t} = \sum_{k=1}^{N_r} \alpha_k(x - v_k) P(x - v_k; t) - \sum_{k=1}^{N_r} \alpha_k(x) P(x; t),$$

(2.14)
2.1. THE CHEMICAL MASTER EQUATION

where \( P(x; t) \) is the probability of \( X(t) = x \) for \( x \in \Omega \) and \( t > 0 \). Hence given a biological/chemical system, whose population is described by (2.12), then the probability distribution over the state space is given by the Chemical Master Equation (CME).

The physical interpretation of the CME (2.14) is as follows: the CME states that the flow of the probability of being at a state \( x \in \Omega \) at time \( t \) is equal to the probability of arriving at \( x \) by reaction \( r_k \), given by \( \alpha_k(x - v_k)P(x - v_k) \), minus the probability of leaving \( x \) by reaction \( r_k \), given by \( \alpha_k(x)P(x; t) \). Considering this over all reactions gives (2.14).

2.1.1 The CME Problem

Our focus is on the construction numerical solvers for the CME (2.14). We present a generalisation of the CME for a chemical or biological system where the population counts are given by the jump Markov process (2.12).

We recall all the parameters of (2.12). Let \( N_s \in \mathbb{N} \) be the number of different species and \( N_r \in \mathbb{N} \) the number of different reactions. Let \( v_k \), for \( k = 1, \ldots, N_r \), be the stoichiometric vectors for the matching reaction index. Let \( V = [v_1, \ldots, v_{N_r}]^T \) be the stoichiometric matrix. Let \( \Omega \subset \mathbb{N}_0^{N_s} \) denote the state space and let \( x_0 \in \mathbb{N}_0^{N_s} \) be the starting population configuration of the system.

To solve for \( P(x; t) \) in (2.14) for \( x \in \Omega \), we define the vector \( p_t \) to be \( (P(x; t))_{x \in \Omega} \) and we define \( dp/dt \) as the vector \( (\partial p(x, t)/\partial t)_{x \in \Omega} \). Then solving the CME is to solve the initial value problem, differential equation,

\[
\left\{ \begin{array}{l}
\frac{dp}{dt} = Ap_t, \quad t > 0, \\
p_0 := (\delta_{x,x_0})_{x \in \Omega}, \quad t = 0.
\end{array} \right.
\]

The state space \( \Omega \) can be infinite. Constructing the solution to (2.15) for the parameters given above is referred to as the CME problem.

From the (2.14) we can deduce that \( A \) satisfies the following conditions:

- \( \sum_{i=1}^{\lvert \Omega \rvert} A_{i,k} = 0 \), for \( k = 1, \ldots, \lvert \Omega \rvert \),

- \( A_{i,i} \leq 0 \) and \( A_{i,j} \geq 0 \) for \( i \neq j \) where \( i, j \in 1, \ldots, \lvert \Omega \rvert \).

In Continuous Time Markov Chain (CTMC) literature, \( A \) is referred to as an infinitesimal generator of the Markov chain [13, 29].

We keep the above notation consistent through the chapters and refer back to this collection of parameters and initial value problem as the CME problem.
There is a change in notation when we refer to elements in the matrix $A$ and elements in the vector $p_t$. When discussing the rows and columns of $A$, we use integer indices. While discussing the vector $p_t$, we index via states ($x \in \mathbb{N}_0^N$). When interchanging notation, we implicitly apply a bijective mapping between the state space, $\Omega$, and the index set of $A$, $J$.

### 2.1.2 Biological Examples

The following are examples of biological problems where the CME is applied. Each example demonstrates a different way that the probability distribution evolves over the state space. The key difficulty in finding the solution of the CME problem is estimating the support of the approximation. In the following examples we can observe the support growing, travelling and bifurcating.

**Example 2.5. Merging modes**

\[
\begin{align*}
R_1 &: A \to B & a_1(A, B) &= 2A \\
R_2 &: B \to A & a_2(A, B) &= B \\
R_3 &: A \to * & a_3(A, B) &= A \\
R_4 &: B \to * & a_4(A, B) &= 0.5B
\end{align*}
\]

Let the initial population of species A and B be $(20, 20)$. See Figure 2.1 for the probability distributions at different time steps. We can see in Figure 2.1 that the support starts in the top left corner, travels down towards the origin in an $L$ shape.

**Example 2.6. Competing clono-types (T-cell Homoeostasis)**

\[
\begin{align*}
R_1 &: * \to A & a_1(A, B) &= \frac{60Ap}{A + B} + \frac{60A(1 - p)}{A + v_1v_2} \\
R_2 &: A \to * & a_2(A, B) &= A \\
R_3 &: * \to B & a_3(A, B) &= \frac{60Bp}{A + B} + \frac{60B(1 - p)}{B + v_1v_2} \\
R_4 &: B \to * & a_4(A, B) &= B,
\end{align*}
\]

where $p = 0.5$, $v_1 = 100$ and $v_2 = 10$. Let the initial population of species A and B be $(10, 10)$. See Figure 2.2 for the probability distributions at different time steps. In Figure 2.2 we see the support actually bifurcates into two non intersecting subsets.
2.1. THE CHEMICAL MASTER EQUATION

Figure 2.1: CME solution of the Merging modes model

Figure 2.2: CME solution of the Competing clono-types (T-cell Homoeostasis) model
Figure 2.3: CME solution of the discrete Infectious Disease (SEIS) model

Example 2.7. Infectious Disease (SEIS)

\begin{align*}
R_1 &: S + I \rightarrow E + I \\
R_2 &: E \rightarrow I \\
R_3 &: I \rightarrow S \\
R_4 &: S \rightarrow * \\
R_5 &: E \rightarrow * \\
R_6 &: I \rightarrow * \\
R_7 &: * \rightarrow S
\end{align*}

\begin{align*}
a_1(S, E, I) &= 0.1SI \\
a_2(S, E, I) &= 0.5E \\
a_3(S, E, I) &= I \\
a_4(S, E, I) &= 0.01S \\
a_5(S, E, I) &= 0.01E \\
a_6(S, E, I) &= 0.01I \\
a_7(S, E, I) &= 0.4
\end{align*}

Let the initial population be \((S, E, I) = (50, 0, 0)\). See Figure 2.3 for the probability distributions at different time steps. In Figure 2.3 we can also see the support growing and stretching over large number of states, where the growth is not symmetric. It is very common for biological problems to grow non-symmetrically.

As we have seen in the examples above, the support of the CME problem is not trivially found. There is currently no algorithm in the literature to estimate how the support will behave for a CME problem a priori. In practise, the support is tackled by considering a big enough hyper-rectangle to capture all possible dy-
namics of the support. However this type of support selection becomes infeasible for large dimensions. This problem is studied further in §2.3.

2.2 CME Structure

In this section we give the explicit solutions to a small class of biological systems and a key result about the operator $e^A$.

Theorem 2.8. For $N \in \mathbb{N}$ let $A \in \mathbb{R}^{N \times N}$ with the property that $\sum_{i=1}^{N}(A)_{i,i} = 0$ for all $j \in \{1, \ldots, N\}$. Also $(A)_{i,j} \geq 0$ for $i \neq j$ and $(A)_{i,i} \leq 0$ for $i = j$. Then

$$(\exp(A))_{i,j} \geq 0 \text{ for all } i, j \in \{1, \ldots, N\}.$$  

Furthermore if $A$ is an infinite matrix, then $\exp(A)_{i,j} \geq 0$ for all $i, j \in \mathbb{N}$.

Proof. Case: ($N < \infty$) using matrix exponential.

Let $\lambda = \min\{(A)_{i,i}, i = 1, \ldots, N\}$. Then we can rewrite $A$ in the following way,

$$A = (A - \lambda I) + \lambda I.$$  

We find $\hat{A} := (A - \lambda I)$ has only non-negative entries. Furthermore $\hat{A}$ and $\lambda I$ are commutative. We see this in the following way,

$$(A\lambda I)_{i,j} = \sum_{m=1}^{N} (A)_{i,m} (\lambda I)_{m,j} = (A)_{i,j}\lambda.$$  

Likewise,

$$(\lambda I A)_{i,j} = \sum_{m=1}^{N} (\lambda I)_{i,m} (A)_{m,j} = \lambda (A)_{i,j},$$  

hence they commute.

We then have

$$e^{\hat{A} + \lambda I} = e^{\hat{A}} e^{\lambda I} = e^{A}.$$  

Since the power series of $e^{\hat{A}}$ has all non-negative entries and also $e^{\lambda I} = I e^{\lambda}$ has all non-negative entries, then $e^{\hat{A}}$ has all non-negative entries. Hence, $e^{A}$ is the product of two matrices, $e^{\hat{A}}$ and $e^{\lambda I}$, with non-negative entries is also non-negative.

Case: ($N = \infty$) uses the framework reaction counts([5.1]).
Let \( I_i \) be the identity vector with one in the \( i \)th column and zero everywhere else. Then \( e^A I_i \) is the solution of the CME problem with initial condition \( I_i \). Let \( p^i \) denote the \( e^A I_i \). Each index, \( j \in \mathbb{N} \), corresponds with a state in the state space and \( p^i_j = (e^A I_i)_j \) is the probability of observing that state. We know from the CME with respect to reaction counts \( \S 5.1 \) that, probability of being a state is the sum of the probability of all the paths that reach that state from the initial state. Since path probabilities are non-negative, we know that \( e^A I_i \) has non-negative column entries, furthermore all columns have non-negative entries.

The columns of \( e^A \) having only non-negative values and summing to be finite give us key results of the Finite State Projection method and its derivatives.

Jahnke et. al showed that for a system with only monomolecular reactions, that is, only one type of molecule is engaged in a reaction, the explicit solution of the CME of such a system can be derived. Table 2.1 gives the list of monomolecular reactions that are commonly depicted in literature.

| Reaction \( r_{jk} \): | \( S_j \xrightarrow{c_{jk}} S_k \) | conversion \((j \neq k)\) |
| Reaction \( r_{0k} \): | \( \ast \xrightarrow{c_{0k}} S_k \) | production from source or inflow |
| Reaction \( r_{j0} \): | \( S_j \xrightarrow{c_{j0}} \ast \) | degradation or outflow |

Table 2.1: Monomolecular reactions.

To state the theorem we introduce the following definitions. Given two probability vector \( p_1 \) and \( p_2 \), we define a convolution, \( p_1 \star p_2 \), of the two probabilities by,

\[
(p_1 \star p_2)(x) = \sum_z p_1(z)p_2(x-z) = \sum_z p_1(x-z)p_2(z),
\]

where \( x,z \in \Omega \).

Let \( N \in \mathbb{N} \) and \( p = (p_1, \ldots, p_N) \in [0, 1]^N \) with \( |p| \leq 1 \). The multinomial distribution \( \mathcal{M}(x, \xi, \nu) \) is given by

\[
\mathcal{M}(x, \xi, \nu) = \begin{cases} 
\xi! \frac{1-|p|^\xi-|x|}{(|x|-|x|)!} \left( \prod_{k=1}^N \frac{\nu_k}{x_k!} \right) & \text{if } |x| \leq \xi \text{ and } x \in \Omega, \\
0 & \text{otherwise}.
\end{cases}
\]

Let \( \mathcal{P}(x, \lambda) \) denote the product Poisson process where,

\[
\mathcal{P}(x, \lambda) = \frac{\lambda^x}{x_1!} \cdots \frac{\lambda^x}{x_n!} e^{\lambda|x|}, x \in \Omega \text{ and } \lambda \in \mathbb{R}_+^N.
\]
Theorem 2.9. [45, Jahnke et. al. Theorem 1]

Consider the monomolecular reactions system, a system with only the reactions from (Table 2.1), with initial distribution \( P(\cdot;0) = \delta_{\xi}(\cdot) \) for some initial population \( \xi \in N^{N_s}_0 \). Let \( c_{ij} \), for \( i,j \in \{1,\ldots,N_s\} \), be the reaction rates for monomolecular reactions given in Table (2.1). For \( t > 0 \), we define \( A(t) \in M^{N_s \times N_s} \) with entries \( a_{ij}(t) \) given by:

\[
a_{ij}(t) := c_{ij}(t) \quad \text{for} \quad i \neq j \geq 1, \quad \text{and} \quad a_{ii}(t) := -\sum_{j=0}^{N_s} c_{ij}(t). \tag{2.16}
\]

Let \( b(t) \in \mathbb{R}^{N_s} \) with entries \( b(t) := (c_{01}(t), \ldots, c_{0N_s}(t)) \).

Then the probability distribution at time \( t > 0 \) is

\[
P(\cdot; t) = P(\cdot, \lambda(t)) \ast M(\cdot, \xi_1, p^{(1)}(t)) \ast \cdots \ast M(\cdot, \xi_{N_s}, p^{(N_s)}(t)). \tag{2.17}
\]

The vector \( p^{(i)}(t) \in [0,1]^{N_s} \) and \( \lambda(t) \in \mathbb{R}^{N_s} \) are solutions of the reaction-rate equations,

\[
\frac{dp^{(i)}(t)}{dt} = A(t)p^{(i)}(t)\quad p^{(i)}(0) = \varepsilon_k,
\]

and

\[
\frac{d\lambda(t)}{dt} = A(t)\lambda(t) + b(t),\quad \lambda(0) = 0.
\]

Using the theorem above, Jahnke et. al. derived equation of motion for the mean and variation for monomolecular systems [45]. These results match with the equations of motion given by Gadgil et. al. who derived their results using moment generating functions [32].

Sunkara in 2009 discussed the analytic solutions for a CME describing a system where the population of any of species cannot decay. That is, the population of the species remain the same or increase. In particular, recursive construction of the analytical solution from the initial state in the state space was provided there. The following proposition is critical in the CME with respect to reaction counts framework, an introduction and discussion is presented in §5.1.

Proposition 2.10. [80, Sunkara Proposition 4]

Let the terms be as in the CME problem constructed in (§2.1.1).

If \( \forall^T \) is invertible and \( P(\cdot;0) = \delta_{x_0}(\cdot) \), then the solution for (2.1.1) is given by,
\[ P(x_0; t) = e^{-(\sum_{j=1}^{N_r} a_j(x_0))t}, \text{ for } t > 0, \tag{2.18} \]

and

\[ P(x; t) = \sum_{i=1}^{N_r} \left( \alpha_i(x - v_j) \int_0^t e^{-(\sum_{j=1}^{N_r} \alpha_j(x))(t-s)} P(x - v_i; s) ds \right), \tag{2.19} \]

for \( x \in \Omega \setminus \{x_0\} \) and \( t > 0 \).

Engblom in 2007 presented that for the finite infinitesimal generator \( A \), the probability at a future time \( t > 0 \) the operator \( e^{At} \) is norm preserving for probability vectors. Furthermore, when \( A \) is infinite then the operator \( e^{At} \) is contractive with Lipsitz constant 1. This is a classical result from semi-groups on Banach spaces \[29\].

**Theorem 2.11.** \[27\] Engblom] Let \( p_0 \in \ell_1 \) have all non-negative values. Let \( A \) be the infinitesimal generator of the CME problem \( (\S 2.1.1) \) and \( t > 0 \).

If \( |\Omega| < \infty \), then

\[ \|e^{At}p_0\|_1 = \|p_0\|_1. \]

If \( |\Omega| = \infty \), then

\[ \|e^{At}p_0\|_1 \leq \|p_0\|_1. \]

Explicit solutions are only known for a small class of problems. In systems of most interest, there are reversible reactions and reactions with multiple species involved. For complex systems, numerical approximations of the CME have to be constructed. The following sections give some of the popular methods for solving the CME problem, \( (\S 2.1.1) \). Our focus is on the theory behind the solvers more so than the implementation.

### 2.3 Finite State Projection Method

In many applications where there are many different species interacting, the complete state space grows very large, in the order of millions. However in this large state space the states which contribute significantly in probability are relatively far fewer. Utilising this sparsity for solving the CME problem\( (\S 2.1.1) \), Munsky and Khammash introduced the Finite State Projection method (FSP) \[49\].

The FSP method is as follows: Let \( A \) be the infinitesimal generator of the CME problem \( (\S 2.1.1) \). Given a tolerance of \( \varepsilon > 0 \), the FSP method determines
2.3. FINITE STATE PROJECTION METHOD

$J$, a subset of the index set of $A$, such that if $A_J$ was a sub matrix of $A$ restricted to $J$, then for $t > 0$

$$\|e^{At}p_0 - e^{A_Jt}p_0\|_1 < \varepsilon.$$ 

Hence the FSP method gives an approximation to the CME problem (§2.1.1) of error $\varepsilon$ by truncating the state space. Furthermore, the approximation generated by the FSP method has the property

$$0 \leq (e^{A_Jt}p_0)_i \leq (e^{At}p_0)_i \leq (e^{A_Jt}p_0)_i + \varepsilon, \text{ for } i \in J.$$ 

The FSP method is algorithmically described in Algorithm 1.

<table>
<thead>
<tr>
<th>Algorithm 1: FSP Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>input</strong>: $p_0, \varepsilon, t$</td>
</tr>
<tr>
<td><strong>output</strong>: $\bar{p}_t$</td>
</tr>
<tr>
<td><strong>begin</strong></td>
</tr>
<tr>
<td>1 find $A_J$, a sub matrix of $A$</td>
</tr>
<tr>
<td>2 $\bar{p}_t \leftarrow e^{A_Jt}p_0</td>
</tr>
<tr>
<td>3 if $|\bar{p}_t|_1 &gt; 1 - \varepsilon$ then</td>
</tr>
<tr>
<td>4 stop</td>
</tr>
<tr>
<td>5 else</td>
</tr>
<tr>
<td>6 Increase $J$ and go to 3</td>
</tr>
<tr>
<td>7 end</td>
</tr>
<tr>
<td>8 return $\bar{p}_t$</td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
</tbody>
</table>

Munsky and Khammash give two results which assist in proving the existence and bounds of the approximation produced via the FSP method.

**Theorem 2.12** ([49, Theorem 2.1]). For $n$ finite. If $A \in \mathbb{R}^{n \times n}$ has non negative off-diagonal elements and columns summing to zero, then for any pair of index sets, $J_2 \supseteq J_1$,

$$[e^{A_{J_2}}]_{J_1} \geq e^{A_{J_1}} \geq 0,$$  \hspace{1cm} (2.20) 

element-wise.

**Theorem 2.13** ([49, Theorem 2.2]). Consider any Markov process in which the probability density evolves according to the linear ODE described in (§2.1.1). Let
$A_J$ be a principle sub matrix of $A$ and let $p_0|_J$ be the restriction of $p_0$ on $J$. If for $\varepsilon > 0$ and $t \geq 0$,
\[
1^T e^{A_J t} p_0|_J \geq 1 - \varepsilon,
\]
then
\[
e^{A_J t} p_0|_J \leq e^{A t} p_0|_J \leq e^{A_J t} p_0|_J + \varepsilon 1,
\]
element-wise.

The Finite State Projection method is a successful method for solving the CME. It was shown to be faster than the conventional stochastic simulation method \cite{35} for computing probability distributions of dimensions up to 30, \cite{59} \cite{42}. Advanced methods to handle stiffness, QSSA \cite{70} \cite{59} and accuracy, Krylov FSP \cite{8} were built on top the FSP method adding to its success.

The following subsection discusses how the approximation error is monitored and controlled in the FSP method.

### 2.3.1 Sink State

The error of the approximation of the FSP method is found by introducing a fictitious state, which is referred to as a sink state. The purpose of this state is to have all the states on the boundary of a finite state space, have their reactions redirected into the sink state. Hence probability from the boundary flowing out of the truncated domain is accumulated into the sink state where it does not leave, see Figure 2.4. Furthermore, the absolute error of the approximation is less then equal to the probability in the sink state.

![Figure 2.4: Flow from domain into the sink state.](image-url)
2.3. FINITE STATE PROJECTION METHOD

Proposition 2.14. Fix $|\Omega| = n \leq \infty$. Let $A \in \mathbb{R}^{n \times n}$ be the infinitesimal generator of the CME problem (2.1.1) and $p_0 \in \mathbb{R}_+^n$ be the initial condition. Fix $J$ to be a subset of the index set of $A$. Let $P_J$ be a permutation matrix which maps $A$ to the form

$$\tilde{A} := P_J A P_J^T = \begin{bmatrix} A_J & B_J \\ C_J & D_J \end{bmatrix}.$$ 

Define $\mathcal{A} \in \mathbb{R}^{|J|+1 \times |J|+1}$ as

$$\mathcal{A} := \begin{bmatrix} A_J & 0 \\ E_J & 0 \end{bmatrix},$$

where

$$(E_J)_{i,j} := \begin{cases} \sum_{m=1}^{|J|} (A_J)_{m,j}, & \text{if } i = 1 \text{ and } j \in 1, \ldots, |J|. \\ 0, & \text{otherwise.} \end{cases}$$

If $\tilde{A}$ is the infitesimal generator of the CME problem (2.1.1), $p_0$ is the initial distribution and $t > 0$, then for $\tilde{A} := \begin{bmatrix} A_J & 0 \\ 0 & 0 \end{bmatrix}$,

$$\| e^{\tilde{A} t} p_0 - e^{\tilde{A} t} p_0 \|_1 = (e^{A_J t} p_0)_{|J|+1}.$$ 

Proof. We consider the power series expansion of $e^{A t}$ given by,

$$e^{A t} = I + t \begin{bmatrix} A_J & 0 \\ E_J & 0 \end{bmatrix} + \frac{t^2}{2} \begin{bmatrix} A_J^2 & 0 \\ E_J A_J & 0 \end{bmatrix} + \frac{t^3}{3!} \begin{bmatrix} A_J^3 & 0 \\ E_J A_J^2 & 0 \end{bmatrix} + \ldots$$

We see that the top left block forms the power series of the matrix $e^{A_J t}$, hence the expression sums to a form,

$$e^{A t} = \begin{bmatrix} e^{A_J t} & 0 \\ Q & 1 \end{bmatrix}.$$ 

For $i \in J$ we have that

$$(e^{\tilde{A} t} p_0)_i = (e^{A_J t} p_0)_i.$$ 

Using this we can write the following,

$$\| e^{A t} p_0 \|_1 = \sum_{i \in J} (e^{A t} p_0)_i + (e^{A_J t} p_0)_{|J|+1}.$$ 

Since $\mathcal{A}$ is of the CME form, we apply Theorem 2.11 to $\| e^{A t} p_0 \|_1$ to get

$$\| p_0 \|_1 = \sum_{i \in J} (e^{A t} p_0)_i + (e^{A_J t} p_0)_{|J|+1}. \quad (2.23)$$
CHAPTER 2. CHEMICAL MASTER EQUATION

Now if we consider

\[ \|e^{\hat{A}t}p_0 - e^{\hat{A}t}p_0\|_1 = \sum_{i \in \Omega} \| (e^{\hat{A}t}p_0)_i - (e^{\hat{A}t}p_0)_i \|, \]

using the positivity from Theorem 2.12 we get,

\[ \|e^{\hat{A}t}p_0 - e^{\hat{A}t}p_0\|_1 = \sum_{i \in \Omega} (e^{\hat{A}t}p_0) - \sum_{i \in J} (e^{\hat{A}t}p_0)_i, \]

applying (2.23) gives us

\[ \sum_{i \in \Omega} (e^{\hat{A}t}p_0) - \sum_{i \in J} (e^{\hat{A}t}p_0)_i = \| \|p_0\|_1 - (\|p_0\|_1 - (e^{\hat{A}t}p_0)_{|J|+1}) \]
\[ = (e^{\hat{A}t}p_0)_{|J|+1}. \]

Hence,

\[ \|e^{\hat{A}t}p_0 - e^{\hat{A}t}p_0\|_1 = (e^{\hat{A}t}p_0)_{|J|+1}. \]

When a CME matrix \( A \) for some subset of its indices \( J \), has the property that \( B_J \), from permuted form

\[ \bar{A} = \begin{bmatrix} A_J & B_J \\ C_J & D_J \end{bmatrix} \]

of \( A \), is a zero matrix, then

\[ (e^{\bar{A}t}p_0)_i = (e^{[A_J]t}p_0)_i, \quad \forall i \in J. \]

An example of such a case is the Poisson process. The \( A \) matrix for the Poisson case can be rearranged to a lower bi-diagonal matrix, which makes the \( B_J \) block have only zero entries. For this particular case the FSP method would give the exact values over the indices in \( J \).

The FSP method states that for any \( \varepsilon > 0 \) there exists a subset of the state space such that, the CME problem (\[2.1.1\]) solved over that subset will give a maximum \( \ell_1 \) global error of \( \varepsilon \). In the literature the \( N \)-step and Sliding Window are the only two methods for growing the domain. We present both these methods.
2.3.2 \textit{N-Step algorithm}

The FSP method Algorithm, Algorithm 1, instructs that if the right error is not attained on the current support, then more states should be added until the approximation attains the right error. This step not addressed enough through the literature, however, it is a very crucial question when considering systems with large particle types. This process of estimating the set of states that should be added is referred to as domain selection. The \textit{N-step} \cite{Munsky2006} and Sliding Window method \cite{Munsky2009} are the adaptive algorithms for domain selection and they are described below.

The \textit{N-step} algorithm finds the subset of the state space for the FSP approximation to be constructed on. It was introduced by Mansky and Khammash alongside the introduction of the FSP algorithm. The \textit{N-step} method working on the logic that the next state to add should be the state which is reached by a reaction from an existing state. Hence the \textit{N-step} method, takes the existing support and generates all the states reachable by one reaction, then all the states reachable by two reactions, and so till the specific \textit{N}. Munsky and Khammash proved that the \textit{N-step} algorithm converges, that is the support of an FSP approximation of error \( \varepsilon \), is the union of all the states reachable by \textit{N} reactions from the initial state, and furthermore \textit{N} is finite.

The \textit{N-step} algorithm is given in Algorithm 2 and the corresponding input parameters are given in Table 2.2.

| \( \Omega \) | current truncated state space. |
| \( N_r \) | number of reactions. |
| \( v_i \) | \( i = 1, \ldots, N_r \), stoichiometric vectors. |
| \( R_i \) | \( i = 1, \ldots, N_r \), reaction map, \( R_i(x) = x + v_i \), for \( x \in \mathbb{N}_0^N \). |

The effectiveness of the \textit{N-step} method for the use of constructing an FSP approximation is discussed in Chapter 5 and numerical implementation and evaluations are given in Section 5.3.

2.3.3 \textit{Sliding Window Algorithm}

The Sliding Window Algorithm (SW) is an FSP based method with a stochastic simulation method for domain selection. In biological/chemical systems, some reactions occur more often than others, which makes the growth of the support
Algorithm 2: $N$-step Algorithm

\begin{verbatim}
input : $\Omega, N_r, v_1, \ldots, v_{N_r}, R_1, \ldots, R_{N_r}$.
output: $\bar{\Omega}$

1 begin
2 \quad $\Omega_0 \leftarrow \Omega$
3 \quad for $i = 1, \ldots, N$ do
4 \quad \quad $\Omega_i \leftarrow \bigcup_{i=1}^{N_r} R_i(\Omega)$ \quad (consider only valid states)
5 \quad end
6 \quad $\bar{\Omega} \leftarrow \bigcup_{i=0}^{N} \Omega_i$.
7 \quad return $\leftarrow \bar{\Omega}$.
8 end
\end{verbatim}

The $N$-step method does not take different firing rates into consideration and grows the domain uniformly. The SW algorithm was introduced by Wolf et. al [88] as a more efficient way for domain selection to the $N$-step algorithm. The SW algorithm is given in Algorithm 3 and the input parameters are given in Table 2.3.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_t$</td>
<td>probability distribution for some time $t$.</td>
</tr>
<tr>
<td>$\Omega_t$</td>
<td>the support of $p_t$</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>time step.</td>
</tr>
<tr>
<td>$N_s$</td>
<td>number of species.</td>
</tr>
<tr>
<td>$N_{SSA}$</td>
<td>number of samples to extract from $\Omega_t$.</td>
</tr>
</tbody>
</table>

The function $\text{Hyper-Rectangle}(x, y)$ outputs a hyper-rectangle with $x$ as the largest co-ordinate and $y$ as the smallest co-ordinate. The function reference SSA is given by Algorithm 4.

The SW algorithm is designed for systems which have a few reactions which fire at a faster rate than the rest of the reactions. This non-uniform reaction firing makes the domain rectangular. The SW method will effectively capture this structure, the SSA simulation component of the algorithm will fire more reactions in the direction of fast firing reactions and give boundaries which form a rectangular shape. Such problems are called stiff problems [40, 72] and these problems cause large domain selections via the $N$-step method. Wolf et. al have shown a significant speed up of computing stiff problems using the SW algorithm.
2.4. STOCHASTIC SIMULATIONS

Algorithm 3: Sliding Window Algorithm

\[
\begin{align*}
\text{input} & : p_t, \Omega_t, \Delta t, N_{SSA}, N_s \\
\text{output} & : \Omega_{t+\Delta t} \\
1 & \text{begin} \\
2 & \quad x_1, \ldots, x_{N_{SSA}} \leftarrow \text{sampled from } \Omega_t \\
3 & \quad h^+ \leftarrow 0 \in \mathbb{N}_0^{N_s} \\
4 & \quad h^- \leftarrow 0 \in \mathbb{N}_0^{N_s} \\
5 & \quad \text{for } i = 1, \ldots, N_{SSA} \text{ do} \\
6 & \quad \quad X = \text{SSA}(x_i, \Delta t) \quad \text{(advance every sample via SSA)} \\
7 & \quad \quad \text{for } j = 1, \ldots, N_s \text{ do} \\
8 & \quad \quad \quad h_j^+ = \max(h_j^+, X_j) \quad (X_j, j\text{th component of } X) \\
9 & \quad \quad \quad h_j^- = \min(h_j^-, X_j) \\
10 & \quad \quad \text{end} \\
11 & \quad \text{end} \\
12 & \quad \Omega_{t+\Delta t} \leftarrow \text{Hyper-Rectangle } (h^+, h^-) \\
13 & \text{return } \Omega_{t+\Delta t} \\
14 & \text{end}
\end{align*}
\]

The SSA component of the SW algorithm gives rise to truncation of modes in multi-modal solutions of the CME. This can be avoided by taking a larger $N_{SSA}$ value. In the current literature there is no rule on the selection of $N_{SSA}$ (the number of SSA simulations). This value would need to grow proportional to the domain.

2.4 Stochastic Simulations

Stochastic Simulation methods are used to construct approximate solutions of the CME problem (\S 2.1.1). An approximation can be constructed by computing multiple realisations using a Stochastic Simulation method, then binning the realisations and dividing by the total number of realisations. Such an approximation is referred to as an empirical distribution. Constructing an approximation for the CME problem via Stochastic Simulations is simple and scalable.

There are numerous stochastic simulation methods in the literature \cite{22,14,9,40,83,3}. The empirical distribution can converge in one of the two ways to the
explicit distribution, the approximation converges weakly or strongly [13]. When an approximation converges weakly, it implies that only the expectation of the empirical distribution (approximation) is converging towards the expectation of the explicit distribution. An empirical distribution is said to converge strongly, if the expectation of the difference between the empirical distribution and explicit distribution is converging to zero. It has been shown by Anderson et. al [2] that the τ-leap method is strongly convergent under certain assumptions.

Most stochastic simulation methods are designed to converge weakly. The intention of the stochastic simulations is to construct realisations as close to the expected behaviour of the stochastic process as possible. Weak convergence is easier to attain and a significant speed up in computation time is achieved [22, 11, 33, 4, 40]. However, our focus is on constructing an empirical distribution for the CME problem (2.1.1), therefore we only focus on strongly converging algorithms. In particular we will be using the Stochastic Simulation Algorithm (SSA) proposed by Gillespie [35], the SSA by construction is an exact simulation of the systems we are interested in [35].

Stochastic simulation methods are widely studied and are very simple to implement. A stochastic simulation realises its support ongoingly and since each realisation is independent to another, the realisations can be parallelised to attain a linear speed up. Hence the question arises

Should the CME be solved by constructing an empirical distribution from stochastic simulations or by approximating the matrix exponential?

When constructing an approximation using the FSP method, the majority of time is spent in finding the support to construct the approximation on. To answer the question on which method to use, we need to compare the number of realisations needed to construct an empirical distribution for a desired error, and the number of points needed in the subset of the state space, to construct an approximate distribution with the same error. In section 4.2 we prove that theoretically, the minimal support of the matrix exponential approximation will always be smaller than the minimal number of realisations needed to construct an empirical distribution of the same error.

We give a brief introduction to the SSA algorithm presented by Gillespie in 1976, followed by the τ-Leap method. Then a brief introduction to the construction of the Quassi Steady State Approximation (QSSA) is given, which we refer to in §2.5.
2.4. STOCHASTIC SIMULATIONS

Stochastic Simulation Algorithm (SSA) [35]

The Stochastic Simulation Algorithm was introduced by Gillespie in 1976 and has since been used as a key tool in constructing simulations in the field of System Biology. Its success inspired faster stochastic simulations, where now, simulations of hundreds of particle types are computed using stochastic simulations [28]. In Algorithm 4 we describe the SSA and Table 2.4 are the list of parameters.

<table>
<thead>
<tr>
<th>Table 2.4: SSA input parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_0$</td>
</tr>
<tr>
<td>$t_0$</td>
</tr>
<tr>
<td>$t_{\text{final}}$</td>
</tr>
<tr>
<td>$N_r$</td>
</tr>
<tr>
<td>$N_s$</td>
</tr>
<tr>
<td>$\alpha_i$</td>
</tr>
<tr>
<td>$v_i$</td>
</tr>
<tr>
<td>$x$</td>
</tr>
</tbody>
</table>

The function \texttt{uniform} returns a sample from the uniform distribution on $[0,1]$. The SSA realisation is constructed by first computing the next time point at which a reaction will occur, which we denote as $t$. Then, the probability that one of the $N_r$ reactions fires is given by the propensity of that reaction divided by the sum of propensities of all the reactions. By progressively computing the next time step at which a reaction occurs and sampling which reaction occurs, a realisation of the underlying jump Markov process is constructed.

$\tau$-Leap

The $\tau$-leap method was introduced by Gillespie in [36] and has been adapted as a more useful stochastic simulation method to the SSA. The $\tau$-leap method functions as follows: you choose a largest time step $\Delta t > 0$ such that in that time step, we can assume that the reactions are firing via a Poisson distribution with constant propensities. Then, within that time step, the realisation is found by sampling from the Poisson distribution to determine which reactions have fired. The difficulty is selecting that largest possible time step $\Delta t$ such that the assumption holds. Please see [36, 3] for the time-step selection criterion. The $\tau$-leap method is more formally known as the Euler approximation of the jump Markov process representation,
Algorithm 4: Stochastic Simulation Algorithm

\textbf{input} : \( x_0, t_0, t_{\text{final}}, \alpha_1, \ldots, \alpha_{N_r}, v_1, \ldots, v_{N_r} \)

\textbf{output}: \( x \)

1 \hspace{1em} \text{begin}
2 \hspace{2em} x \leftarrow x_0
3 \hspace{2em} t \leftarrow t_0
4 \hspace{2em} \textbf{while} \; t < t_{\text{final}} \hspace{1em} \textbf{do}
5 \hspace{3em} \alpha_0 \leftarrow \sum_{i=1}^{N_r} \alpha_i(x)
6 \hspace{3em} \textbf{if} \; \alpha_0 == 0 \hspace{1em} \textbf{then}
7 \hspace{4em} t \leftarrow t_{\text{final}}
8 \hspace{4em} \textbf{break}
9 \hspace{3em} \textbf{end}
10 \hspace{3em} r_1, r_2 \leftarrow \text{uniform} \; (0, 1)
11 \hspace{3em} \tau \leftarrow \frac{1}{\alpha_0} \log \left( \frac{1}{r_1} \right)
12 \hspace{3em} \textbf{if} \; t + \tau > t_{\text{final}} \hspace{1em} \textbf{then}
13 \hspace{4em} t \leftarrow t_f
14 \hspace{4em} \textbf{break}
15 \hspace{3em} \textbf{end}
16 \hspace{3em} t \leftarrow t + \tau
17 \hspace{3em} \text{choose } j \text{ such that } \sum_{i=1}^{j-1} \alpha_k(x) < \alpha_0 r_2 \leq \sum_{i=j}^{N_r} \alpha_i(x)
18 \hspace{3em} x \leftarrow x + v_j
19 \hspace{2em} \textbf{end}
20 \hspace{2em} \textbf{return} \; x
21 \hspace{1em} \textbf{end}

\[ X_{N_r}(t) = X_{N_r}(0) + \sum_{k=1}^{N_r} Y_k \left( \int_0^t \alpha_k(X_{N_r}(s)) ds \right) v_k. \] \hspace{1em} (2.24)

The approximation is given by: Let \( \tau_0 = 0, \ldots, \tau_n = t_{\text{final}} \) be \( n + 1 \) points in the interval \([0, t_{\text{final}}]\). Let \( \tilde{X}_{N_r}(\tau_0) = x_0 \) the initial state of the system.

For \( i = 1, \ldots, n \), the \( \tau \)-leap realisation is given by

\[ \tilde{X}_{N_r}(\tau_i) = \tilde{X}_{N_r}(\tau_0) + \sum_{k=1}^{N_r} Y_k \left( \sum_{j=1}^{i-1} \alpha_k(\tilde{X}_{N_r}(\tau_j))(\tau_{j+1} - \tau_j) \right) v_k. \]

Then we simple sample from a Poisson distribution to find out how many reactions have occurred. For more details on implementation please refer to the


2.4. STOCHASTIC SIMULATIONS

survey \[36, 3\].

QSSA

Rao and Atkins described a splitting of the reactions into fast and slow reactions \[72\], where the fast reactions had propensities which were several orders of magnitude larger than the slow propensities. These fast reactions are considered to reach equilibrium, hence Rao and Atkins proposed computing the fast reactions separately and substituting the steady state of the fast reactions into the computation of the slow reactions. Rao and Atkins stated that if the fast moving state space does reach steady state, then the approximate fast and slow stochastic processes are still individually Markovian. This proposition of splitting reactions, led to the development of multi-scale stochastic simulation algorithms \[72, 40, 4\], decreasing the computation complexity of many biological problems. The mathematical formulation is given below.

Let \( R_f \) be the set of indices of the fast reactions, likewise, let \( R_s \) be the set of indices of the slow reactions. Let \( N_{f,r} = |R_f| \) and \( N_{s,r} = |R_s| \).

\[ R = R_f \cup R_s \]

Then

\[ X_f(t) = X(0) + \sum_{k \in R_f} Y_k \left( \int_0^t \alpha_k(X_f(s)) ds \right) v_k \quad (2.25) \]

The assumption is that the fast reactions reach steady state rapidly (Quasi-steady state), this is denoted by \( X_f(\infty) \). This steady state is then substituted into the stochastic process governing the slow reactions to give,

\[ X_s(t) = X(0) + \sum_{k \in R_s} Y_k \left( \int_0^t \alpha_k((X_f(\infty), X_s(s))) ds \right) v_k \quad (2.26) \]

Through the last decade, a variation of approximations like the QSSA have been introduced to minimise the computation time of realisations.

Another approximation is the Hybrid method, where a species in the system is considered to be governed by deterministic equations, then the solution of the deterministic equations are substituted into the stochastic processes \[42, 47, 63\].

It is important to note that after the reductions have been taken into account, there is still a portion of the problem which is non-reducible. This non-reducible part of the system has to be computed conventionally, Chapters \[3\] and Chapter \[4\] give the framework in which the irreducible parts can be efficiently computed.
Considering the new framework given in Chapter 3 and Chapter 4, the class of problems considered computable can be moved into the tens of dimensions.

## 2.5 Aggregation

The FSP method states that there exists a finite subset of the state space on which to construct an approximation for a prescribed error. In high dimensional problems (large number of species) the solution of the CME (§2.1.1) has large support, in the order of millions. The solution of the CME problem is a probability mass function over the state space, hence the sum of the probability over the state space is one.

This combination of large support and approximation summing to one gives rise to large subsets of state space having very small variations among adjacent states. If the variation between adjacent states is very small, then a natural question arises of whether it is possible to aggregate portions of the state space to simplify computation [42, 31, 59]. To tackle systems with large state spaces, the theory of aggregation for the CME was introduced to help reduce the complexity of the FSP approximation. The process of aggregation can be split into two steps: the first step is to aggregate the state space which gives a simpler CME problem, the second step is to take the solution of the aggregated CME problem and disaggregate to construct an approximation over all state space. Below, the two steps of aggregating the CME problem are discussed. We only present the framework of the aggregation methods, for solver and implementation see [59, 42]. In Chapter 3, we discuss how the new FSP framework can help advance the aggregation methods.

From the CME problem (§2.1.1), let $A$ be the infinitesimal generator, $\Omega$ the state space and $p_t$ the probability mass function over $\Omega$ at time $t$. To aggregate is to partition $\Omega$ into non-intersecting subsets. Let $n_a$ be the number of subsets $\Omega$ will be aggregated into. If we denote $\Omega^i$ as the $i$th subset and $p^i_t$ defined by,

$$p^i_t = p_t(\Omega^i) := \sum_{x \in \Omega^i} p_t(x).$$

The aggregation operator for this partitioning is denoted by $E$, $E \in \mathbb{N}_0^{n_a \times |\Omega|}$ and is given by

$$E_{i,j} = \begin{cases} 1, & \text{if } j\text{th state is in } \Omega^i, \\ 0, & \text{otherwise}. \end{cases}$$
2.5. AGGREGATION

The aim is to solve for a $\tilde{p}_t \in \mathbb{R}_+^{n_a}$ given by

$$\tilde{p}_t = Ep_t.$$  

For $\tilde{p}_t$ to be of the CME problem form, there must exist a $B \in \mathbb{R}^{n_a \times n_a}$ such that

$$EA = BE$$

where $A$ is the infinitesimal generator of the CME problem (§2.1.1). There are different ways to select the partitioning for aggregation. Zhang, in 2009, [43] proposed an adaptive aggregation method by running some stochastic simulations and having coarser aggregation where the realisations didn’t occur a lot and finer aggregation where many realisations occurred.

Hegland et. al. in a paper [42] proposed aggregating on a fixed grid and aggregating more in the direction of the state space growth. Most recently MacNamara et. al. introduced a method which aggregates all the Quasi-steady state features of a system, reducing the CME problem to only needing to be computed on the slowest time scale, giving a significant speed up in computation [59].

After reducing the problem to be only $n_a$ states in size, the next step is to take the approximation constructed by the aggregation and disaggregate to have an approximation over the complete state space $\Omega$. In general the disaggregating operator is denoted by $F$. The aim is to construct the operator $F$ such that $FE = I$, where $I$ is the identity matrix. If the disaggregator satisfied $FE = I$ then it would be a perfect reconstruction,

$$F\tilde{p}_t = FEp_t = p_t.$$  

A particular choice of $F$ is given by

$$F := E^T(EE^T)^{-1}.$$  

The disaggregation can be considered as an interpolation operator. L"otstedt et. al proposed an polynomial interpolation as a disaggregation operator [31] for low dimensional problems. Hegland et al proposed Sparse Grid construction [42] for the disaggregation. MacNamara et. al for the QSSA method [59] proposed finding $F$ such that $FE$ would equal the Qassi-stationary distribution. It is important to note that if the aggregation operator, $E$, is chosen such that there does not exist a $B$ satisfying $EA = BE$, then the disaggregation will always be an approximation and a complete reconstruction would not be possible.
Aggregation reduces particular features of the CME problem, however the reduced problem is another CME problem in the QSSA method [59]. Therefore the FSP method needs to be used to compute the non reducible part of the problem. In Section 3, 4, 5 new framework for the FSP methods are proposed. We conjecture that the proposed algorithms can be applied to the existing aggregation literature and help expand the class of problems they can compute.

2.6 The \( \ell_2 \) Projections of the CME

To tackle the difficulty of computing the solution of the CME problem (§2.1.1) for systems with multiple particle types, advanced methods were introduced to reduce computation and attain the same accuracy [44, 41, 16, 25]. The basic idea of the methods is to project the CME problem (§2.1.1) onto a finite dimensional Hilbert space, representing the solution as a sum of orthonormal basis functions which span the Hilbert space. The new problem is then to find the weights of the basis function to most accurately reconstruct the solution of the CME (§2.1.1). We refer to the class of methods used to solve this new problem as the \( \ell_2 \) projections of the CME.

The motivation to project the CME problem (§2.1.1) onto a Hilbert space came from the observations that the coefficients of the basis functions decay faster than the probabilities over the state space. If the solution of the CME problem (§2.1.1) was smooth enough and decayed fast, the number of basis functions needed to construct the solution would far fewer and simpler to compute. In \( \ell_2 \) projection methods, we assume that the state space is finite. This ensures that the required number of basis weights is always at most as same as the size of the state space.

Below we derive a generalisation of the framework, described by Jahnke et. al [16], to map the CME problem onto \( \ell_2 \). Concurrently and independently to Jahnke et al., Engblom [25] proposed a similar framework but with a different class of basis functions. For information on numerical schemes and implementation of the CME problem in the \( \ell_2 \) see [46, 25].

Let \( \Omega \) be the state space of the CME problem (§2.1.1). Then, let \( \bar{\Omega} \) be a finite subset of \( \Omega \) which is large enough that the probabilities and their derivatives are close to zero. Let \( \ell_2(\bar{\Omega}) \) be an finite dimensional Hilbert space with a Euclidean point-wise inner product \( \langle \cdot, \cdot \rangle \). Let \( \{\psi_i, i = 1, \ldots, N_b\} \) be the set of orthonormal basis of \( \ell_2(\bar{\Omega}) \), where \( N_b \) is the cardinality of the set. Then for \( f \in \ell_2(\bar{\Omega}) \),
2.6. THE $\ell_2$ PROJECTIONS OF THE CME

\[ f(x) := \sum_{i=1}^{N_b} (\psi_i, f) \psi_i(x), \forall x \in \bar{\Omega}. \]  
\[ \tag{2.27} \]

Given an orthonormal set, we define

\[ \Psi : \ell_1(\bar{\Omega}) \to \ell_2(\bar{\Omega}) \text{ as } \Psi(f) := ((f, \psi_i))_{i=1}^{N_b} \text{ for } f \in \ell_1(\bar{\Omega}). \]  
\[ \tag{2.28} \]

Likewise, in the opposite direction, define

\[ \Psi^{inv} : \ell_2(\bar{\Omega}) \to \ell_1(\bar{\Omega}) \text{ as } \Psi^{inv}(a)(x) := \sum_{i=1}^{N_b} a_i \psi_i(x), \]  
\[ \tag{2.29} \]

for $x \in \bar{\Omega}, a \in \ell_2(\bar{\Omega})$. If $f \in \ell_1(\bar{\Omega})$, then

\[ f = \Psi^{inv}(\Psi(f)). \]

Since the CME problem evolves over time, consider the representation that for $t > 0$ and $x \in \bar{\Omega}$ the solution $P(x; t)$ of §2.1.1 is of the form

\[ P(x; t) = \sum_{i=1}^{N_b} a_i(t) \psi_i(x), \]  
\[ \tag{2.30} \]

for some $a(t) \in \ell_2(\bar{\Omega})$.

Then, multiplying [2.14] by test functions $\psi_i$ and integrating over $\bar{\Omega}$ gives the variational formulation, with the derivation given below. Recall that the derivative of the probability being in some state $x$ at time $t$ is given by

\[ \frac{\partial P(x; t)}{\partial t} = \sum_{k=1}^{N_r} \alpha_k(x - v_k)P(x - v_k; t) - \sum_{k=1}^{N_r} \alpha_k(x)P(x; t), \]

and considering over all $x$ we have,

\[ \frac{dp_t}{dt} = Ap_t. \]

Let $\psi_i$ be a basis function of the orthonormal set $\{\psi_i \in \ell_2(\bar{\Omega}), i = 1, \ldots, N_b\}$, then

\[ \frac{\partial \sum_{i=1}^{N_b} a_i(t) \psi_i(x)}{\partial t} = \sum_{k=1}^{N_r} \alpha_k(x - v_k) \sum_{i=1}^{N_b} a_i(t) \psi_i(x - v_k) - \sum_{k=1}^{N_r} \alpha_k(x) \sum_{i=1}^{N_b} a_i(t) \psi_i(x). \]

Fix $\psi_j \in \{\psi_i \in \ell_2(\bar{\Omega}), i = 1, \ldots, N_b\}$ and multiply it to the equation above,
\[
\frac{\partial}{\partial t} \sum_{i=1}^{N_b} a_i(t) \psi_i(x) \psi_j(x) = \sum_{k=1}^{N_r} \alpha_k(x - v_k) \sum_{i=1}^{N_b} a_i(t) \psi_i(x - v_k) \psi_j(x) - \sum_{k=1}^{N_r} \alpha_k(x) \sum_{i=1}^{N_b} a_i(t) \psi_i(x) \psi_j(x).
\]

Integrating over \( \Omega \) gives,

\[
\int_{x \in \Omega} \left( \frac{\partial}{\partial t} \sum_{i=1}^{N_b} a_i(t) \psi_i(x) \psi_j(x) \right) = \int_{x \in \Omega} \left( \sum_{k=1}^{N_r} \alpha_k(x - v_k) \sum_{i=1}^{N_b} a_i(t) \psi_i(x - v_k) \psi_j(x) - \sum_{k=1}^{N_r} \alpha_k(x) \sum_{i=1}^{N_b} a_i(t) \psi_i(x) \psi_j(x) \right).
\]

Interchanging the sum and integrand and then applying the orthogonality property of the basis functions, the above equation reduces to

\[
\frac{\partial a_j(t)}{\partial t} = \sum_{i=1}^{N_b} \sum_{k=1}^{N_r} \int_{x \in \Omega} (a_i(t) \alpha_k(x - v_k) \psi_i(x - v_k) \psi_j(x) - a_i(t) \alpha_k(x) \psi_i(x) \psi_j(x)).
\]

By considering over all \( j = 1, \ldots, N_b \) we get the matrix initial value problem

\[
\frac{da(t)}{dt} = Aa(t),
\]

for \( t > 0 \) and initial condition \( \Psi(p_0) \), here \( p_0 \) is the initial condition of the CME problem (§2.1.1). The matrix \( A \) is called the Galerkin matrix and is given by

\[
A_{ij} = \begin{cases} 
\sum_{k=1}^{N_r} \int_{x \in \Omega} \alpha_k(x - v_k) \psi_i(x - v_k) \psi_j(x) & i \neq j, \\
\sum_{k=1}^{N_r} \int_{x \in \Omega} \alpha_k(x - v_k) \psi_i(x - v_k) \psi_j(x) - \sum_{k=1}^{N_r} \alpha_k(x) & i = j.
\end{cases}
\]

Note that \( A \) is not Markovian. The infinitesimal generator of the CME problem (§2.1.1), \( A \), is a very sparse matrix where as the Galerkin matrix, \( A \), can be dense based on the basis functions chosen.

The reason for solving this initial value problem rather than the CME problem (§2.1.1) is that, \( a(t) \) decays faster than \( p_t \) for any given \( t > 0 \). This has not been proven, but empirical results show that \( a(t) \) decays fast giving a smaller problem to compute \( \ell_2 \). Let us consider a simple example to demonstrate how and why the \( \ell_2 \) projection methods might be useful. Let us consider a Poisson distribution in one dimension. For different time steps, we compare the smallest support for an error of 0.05 in the \( \ell_1 \) norm and the smallest number of Haar basis functions needed to construct an approximation for an identical error of 0.05. We present the values at different time points in Figure 2.5. The figure shows that the minimal support grows polynomially as expected, however the number of Haar
basis functions needed to construct an approximation of 0.05 stays flat. The reason is that as the time steps increase, the shape of the Poisson distribution becomes wider and flatter. This is reflected in the basis functions, since, as the function gets wider, the Haar basis which reconstructs the distribution just moves up a resolution level. By moving up a resolution level, we have the same shape and nearly the same number of basis functions, which is the reason why the number of basis functions in Figure 2.5 stays flat.

Engblom formulates [25] a similar variational problem as described above, however the chosen basis elements are moving weight functions which need a weighted inner product for orthogonality.

Other $\ell_2$ based approximations are being developed based on the theory of multi-linear algebra [52, 67]. The idea is to use low rank tensors on a manifold to find solutions to the CME problem (§2.1.1). Jahnke et. al presented a general theory for low-rank approximations [44]. More recently Hegland and Garcke [41] presented a rank-1 approximation of the CME.

The $\ell_2$ projections of the CME are not discussed further in this thesis. However, the best $N$-term approximation framework constructed to prove the Optimal Finite State Projection method can be utilised to prove a key question in regards to Galerkin approximation of the CME. There is evidence that when considering the Haar basis functions, the number of coefficients required to construct an approximation of a particular error, is significantly less than the size of the support of the best $N$-term approximation of the same error.
Figure 2.5: Number of Haar wavelet basis coefficients and minimal support size of the OFSP, required to construct an approximation of the Poisson distribution with error 0.05 in the $\ell_1$ norm, at different time steps.
Chapter 3

Optimal Finite State Projection Method

The Optimal Finite State Projection (OFSP) method guarantees an optimal order support size onto which the CME problem can be projected.

The OFSP method is an extension of the Finite State Projection method (FSP) with respect to support size guarantee. Theorems 2.13 and 2.12 state that there is a minimal support that the CME problem can be projected onto for a prescribed error. However, the formulation of the FSP algorithm does not guarantee this without an exhaustive search through the domain. These exhaustive searches for the minimal support become computationally unfeasible as the CME problem evolves over time. Two methods were considered to estimate the domain. The first is an a priori thresholding which truncates a fixed amount at each time step to delete states. The second is a method where a hyper-rectangle is constructed to capture all the probabilities. However, both these methods have flaws. The thresholding method removes newly added states. Furthermore, the threshold for the truncation has to be a small enough value so as to not significantly effect the error of the approximation. This results in almost no significant reduction in domain size. The second method of fixed domain also results in a large domain. The size of the domain is the product of the maximum number of particles in each dimension. Hence, problems with 10 or more particles and with 6 or more particle types, become unfeasible to compute. The OFSP method extends on the FSP method by introducing a simple compression after each time step. This method is proven to give us a domain size which is of optimal order.

The problem of the growing support arises because an element in the state
space is representative of a possible population configuration which the system can be in. As time passes, the number of population configurations that the system can be in grows as we take the combinations of possible events. The support of the CME problem grows rapidly and it becomes computationally expensive to compute and track individual states. This growth is accounted for in the FSP algorithm. However, with it’s two methods of fixed domain and thresholding domain selection, it is claimed to be slow and so approximation methods using QSSA \cite{38,39}, Galerkin \cite{24,46}, and aggregations \cite{42} were introduced to supersede it. These methods were successfully shown to be faster than the FSP method, but their algorithmic complexity still resulted in slow computation times \cite{8,42} for larger problems. Sunkara and Hegland address the domain selection issue of the FSP and introduce the OFSP method \cite{81}. Adding a compression step at the end of a time step gives a significant speed up and reduces the support size to the order of the minimal support size. We see in section \ref{sec:3.3} that if we recompute the problems in which the FSP was shown to be not tractable, we find that the OFSP is faster than the aforementioned advanced approximation methods.

In this Chapter we describe the OFSP method as first presented by Sunkara and Hegland \cite{81}. Furthermore, we derive the best $N$-term approximation theory in extensive detail to prove that the OFSP method gives us a support size that is of the order of the minimal support. We compute some classical examples which are shown by earlier literature to be slow when using the FSP method. We compare the computation times with the methods which replaced the FSP method. We demonstrate that the FSP method is still a contender as a fast and simple method for computing large CME systems.

### 3.1 Best $N$-term approximation

The best $N$-term approximation of a sequence is an approximation with the smallest possible support \cite{17}. The motivation for exploring this paper comes from its use in adaptive wavelet methods \cite{11,12}. In this work, best $N$-term approximation and non-linear approximation theory are used to construct a framework within which it is shown that methods in basis selection were of optimal order. That is, the number of basis functions generated by the adaptive wavelet formulation is to the order of the minimal basis. For an adaptive method, this is a significant result since it shows that the construction of the approximation
is optimised based on memory size and decreases its computation time by reduc-
ing unnecessary computation. Dahmen et. al. gave a benchmark through error bounds to compare different adaptive methods \cite{11}. This same concept was adapted by Sunkara and Hegland to use the best $N$-term approximation, now in the $\ell_1$ setting, as the benchmark for the optimal support size for the approximation \cite{81}. The idea is to be able to compare the supports of different approximation methods for the CME by studying how close they are to the support of the best $N$-term approximation.

In this section we use the weak $\ell_r$ Lorentz spaces \cite{17} to derive the bounds for the support of the best $N$-term approximation in the $\ell_1$ setting. We start by introducing the best $N$-term approximation in the $\ell_1$ setting, then introduce the approximation space and the weak $\ell_r$ space. The inequality in norm between these two spaces gives us, as a corollary, the bounds of the support for the best $N$-term approximation.

Let $P_N : \ell_1 \to \ell_1$ be a projection with the following properties: for $v \in \ell_1(\Omega)$

I) $|\text{supp } P_N v| = N$,

II) there exists $\theta_N > 0$ such that $(P_N v)_i > \theta_N$ for all $i \in \text{supp } P_N v$.

We call $P_N v$ the best $N$-term approximation of $v$.

We take the work by Dahmen et al \cite{11} as motivation to describe the support of the best $N$-term approximation with respect to the $\ell_1$ norm.

3.1.1 $A^s$ Spaces, $s > 0$.

We consider the following spaces. For $N = 1, 2, \ldots$, define

$$\Sigma_N := \bigcup_{i=0}^{N} \{v \in \ell_1(\Omega) : |\text{supp } v| = N\},$$

a non-linear subspace of $\ell_1(\Omega)$ consisting of all vectors with at most $N$ non-zero coordinates. We now characterise vectors $v \in \ell_1(\Omega)$, which can be efficiently approximated by the elements of $\Sigma_N$. For $s > 0$, we define $A^s$ to denote the set of all vectors $v \in \ell_1(\Omega)$ such that the quantity

$$\|v\|_{A^s} := \sup_{N \in \mathbb{N}_0} (N + 1)^s \sigma_N(v),$$

(3.1)
where
\[ \sigma_N(v) := \inf_{w \in \Sigma_N} \| v - w \|_1, \]
is finite. We call \( A^s \) an approximation space [17].

**Lemma 3.1.** For \( v \in \ell_1(\Omega) \), if \( v \in A^s \) for \( s > 0 \), then
\[ \| v - P_Nv \|_1 \leq \| v \|_{A^s} N^{-s}. \]  
(3.2)

**Proof.** Fix \( N \in \mathbb{N} \). By definition of the best \( N \)-term approximation \( \sigma_N(v) = \| v - P_Nv \|_1 \). If \( v \in A^s \), then \( \| v \|_{A^s} \geq (N + 1)^s \sigma_N(v) \). Rearranging this gives us the inequality above.

Therefore \( A^s \) consists of vectors which can be approximated with order \( O(N^{-s}) \) by the elements of \( \Sigma_N \).

The Lemma above states that the best \( N \)-term approximation of an element in \( \ell_1(\Omega) \) has error of \( O(N^{-s}) \). We introduce the weak \( \ell_r \) space, denoted by \( \ell_{r,\infty} \), which is the space of sequences, rearranged in descending order, that decay at \( O(n^{-1/r}) \). Bridging \( \ell_{r,\infty} \) and \( A^s \) will give us the order of the support of the best \( N \)-term approximation. We now derive the foundation of the weak \( \ell_r \) space, where \( 0 < r < 1 \).

### 3.1.2 Weak \( \ell_r \) spaces, \( 0 < r < 1 \).

We introduce the non-increasing rearrangement, \( v^* \), of a vector \( v \). That is, there exists an enumeration \( \tau : \mathbb{N} \to \Omega \) such that the sequence \( v^* \) defined by \( v^*_k = |v_{\tau(k)}| \) is non-increasing, i.e., \( v^*_k \geq v^*_{k+1}, \ k \in \mathbb{N} \). We let \( \ell_{r,\infty} \) be the subspace of all vectors in \( \ell_1(\Omega) \) such that the quantity
\[ |v|_{r,\infty} := \sup_{n \in \mathbb{N}} n^{1/r} v^*_n \]  
(3.3)
is finite. Note that \( | \cdot |_{r,\infty} \) is only a quasi-norm as it does not satisfy the triangle inequality. The \( \ell_{r,\infty} \) space under the \( | \cdot |_{r,\infty} \) quasi-norm is called a weak \( \ell_r \) space. This is a special case of a Lorentz sequence space [18].

The proposition and lemma below give the key properties of the \( \ell_{r,\infty} \) space needed for our analysis.

**Proposition 3.2.** For \( r \in (0, 1) \) and \( v \in \ell_1(\Omega) \). The following three statements are equivalent:
3.1. BEST N-TERM APPROXIMATION

1. \( v \in \ell_{r,\infty} \)

2. There exists \( M > 0 \) such that the non-increasing rearrangement \( v^* \) of \( v \) satisfies \( v^*_k \leq Mk^{-1/r} \), for all \( k \in \mathbb{N} \).

3. There exists an \( M > 0 \) such that for all \( \theta > 0 \),

\[
|\{ x \in \Omega : |v_x| > \theta \}| \leq M^r \theta^{-r}.
\]

Furthermore the smallest \( M > 0 \) for which the bounds above are satisfied is \( M = |f|_{r,\infty} \).

Proof. Fix \( r \in (0,1) \) and let \( M := |v|_{r,\infty} \).

\((1 \to 2)\) : Fix \( v \in \ell_{r,\infty} \). By definition of the \( \ell_{r,\infty} \) norm, (3.3), we get

\[
|v|_{r,\infty} = M \geq n^{1/r} v^*_k, \text{ for all } k \in \mathbb{N}.
\]

Rearranging the inequality gives us the result.

\((2 \to 3)\) : For a vector \( v \) we assume that \( v^*_k \leq Mk^{-1/r} \) for some \( M > 0 \). Then for some threshold \( \theta > 0 \), there is a smallest \( N \) such that \( N^{-1/r} M \geq \theta \), and

\[
N > |\{ x \in \Omega : |v_x| > \theta \}|.
\]

By rearranging the powers and making \( N \) the subject, we find

\[
|\{ x \in \Omega : |v_x| > \theta \}| \leq M^r \theta^{-r}.
\]

\((3 \to 1)\) : Fix \( K \in \mathbb{N} \) such that \( K = |\{ x \in \Omega : |v_x| > \theta \}| \leq M^r \theta^{-r} \). Then \( v^*_K \leq \theta \). After rearrangement we have \( v^*_K \leq MK^{-1/r} \). From definition (3.3), we have \( |v|_{r,\infty} \leq M \). Hence \( v \in \ell_{r,\infty} \).

A consequence of the proposition above is that for any \( v \in \ell_{r,\infty} \), we can see that the rate of decay of \( v^* \) is \( O(n^{1/r}) \). Our aim is to link the weak \( \ell_r \) spaces into the approximation spaces \( A^s \), so that we can describe the best \( N \)-term approximation in terms of \( \ell_{r,\infty} \) norm. This link is a significant result in non-linear approximation theory [17].

The \( \ell_{r,\infty} \) quasi-norm has a monotonic property which we derive and utilise further in this section.

Lemma 3.3. For \( r \in (0,1) \). Let \( v, u \in \mathbb{R}_+^\mathbb{N} \) be two probability distributions such that \( u \geq v \), that is

\[
u_x \geq v_x, \text{ for all } x \in \Omega.
\]

Then

\[
u^*_k \geq v^*_k, \text{ for all } k \in \mathbb{N}.
\]
Furthermore,
\[ |u|_{r,\infty} \geq |v|_{r,\infty}. \]

**Proof.** For \( u, v \in \ell_{r,\infty} \) with \( r \in (0, 1) \), if \( u_x \leq v_x \) for all \( x \in \Omega \). Then
\[ u^*_k \leq v^*_k \text{ for all } k \in \mathbb{N}. \] (3.4)

\[
|u|_{r,\infty} := \sup_{k \in \mathbb{N}} u^*_k k^{1/r},
\leq \sup_{k \in \mathbb{N}} v^*_k k^{1/r}, \quad \text{applying (3.4),}
= |v|_{r,\infty}.
\]

**Remark 3.4.** What \( r \) and \( s \) are for the CME problem is an open problem. For even a simple system such as a birth-death process it is not too clear. If this could be answered, we can then use this result to show that particular infinite dimensional CME problems, (that is state space is infinite in size) induce a strongly continuous generator. For the finite state space case, the CME problem is trivially in all the spaces.

### 3.1.3 Best \( N \)-term approximation bound

We give the relationship between \( \ell_{r,\infty} \) and \( \mathcal{A}^s \). We want the construction to be specific for the CME so that we can get direct values for the coefficients which we use later. The focus is thus directly on probability vectors in \( \ell_1 \). Let \( X \) be the set in which the solutions of the CME exist, that is
\[
X := \{ v \in \ell_1(\Omega) : \forall i \in \mathbb{N} \; 0 < v_i < 1 \text{ and } \|v\|_1 = 1 \} \quad (3.5)
\]

**Theorem 3.5.** Fix \( v \in X \). Let \( r \in (0, 1) \) and \( s \) satisfying
\[
\frac{1}{r} = s + 1.
\]
If \( v \in \ell_{r,\infty} \), then \( v \in \mathcal{A}^s \) and
\[
\|v\|_{\mathcal{A}^s} \leq C_2 |v|_{r,\infty}, \text{ for some } C_2 > 0.
\]
3.1. BEST N-TERM APPROXIMATION

Proof. Let \( v \in \ell_{r,\infty} \), with \( v^*_k = |v|_{r,\infty} k^{-1/r} = |v|_{r,\infty} k^{-s-1} \).

Then

\[
\|v - P_N v\|_1 = \sum_{N+1}^{\infty} v^*_k \\
\leq |v|_{r,\infty} \left( \frac{1}{(N+1)^{s+1}} + \sum_{k=N+2}^{\infty} \frac{1}{k^{s+1}} \right) \\
\leq |v|_{r,\infty} \left( \frac{1}{(N+1)^{s+1}} + \int_{N+1}^{\infty} \frac{1}{k^{s+1}} dk \right) \\
\leq |v|_{r,\infty} \left( \frac{1}{(N+1)^{s}} + \frac{(N+1)^{-s}}{s} \right) \\
\leq |v|_{r,\infty} \left( \frac{1}{s+1} \right) (N+1)^{-s}
\]  

(3.7)

To show \( \|v\|_{A^s} \) is finite, consider

\[
\|v\|_{A^s} = \sup_{N \in \mathbb{N}_0} (N+1)^s \sigma_N(v) \\
= \sup_{N \in \mathbb{N}_0} (N+1)^s \|v - P_N v\|_1 \\
\leq \sup_{N \in \mathbb{N}_0} (N+1)^s |v|_{r,\infty} \left( \frac{1}{s+1} \right) (N+1)^{-s} \\
\leq \left( \frac{1}{s+1} \right) |v|_{r,\infty}.
\]  

(3.8)

Hence for \( r \in (0, 1) \) and \( s = 1/r - 1 \), if \( v \in \ell_{r,\infty} \), then \( v \in A^s \). Furthermore, the claimed bound holds with \( C_2 = \left( \frac{1}{s+1} \right) \).

The proof for the second half of the norm equivalency is not complete, the initial proposed proof was found to be incorrect. The norm equivalency between the weak \( \ell_r \) space and the \( A^s \) approximation space has been proven in the \( \ell_2 \) case by DeVore [17]. There was a failed attempt to translate the proof for the \( \ell_2 \) case for the \( \ell_1 \). Hence, we leave this result as a conjecture and give some notes on what could be the direction to go about proving it.

**Conjecture A.** Fix \( v \in X \). Let \( r \in (0, 1) \) and \( s \) satisfying

\[
\frac{1}{r} = s + 1.
\]
If \( v \in \ell_{r,\infty} \), then \( v \in A^s \) and
\[
|v|_{r,\infty} \leq C_1 \|v\|_{A^s}
\]
for \( C_1 > 0 \).

**Proof.** Fix \( v \in A^s \) for \( s = 1/r - 1 \) and \( r \in (0, 1) \). For \( n = 1, 2, \ldots \),
\[
v_{2n}^* \leq \frac{1}{n} \sum_{k=n+1}^{2n} v_k^*,
\]
\[
\frac{1}{n} \|v - P_n v\|_1,
\]
\[
\frac{1}{n} \|v\|_{A^s} n^{-s},
\]
\[
= \|v\|_{A^s} n^{-s-1}.
\]

We have \(-s - 1 = 1/r\). Then
\[
v_{2n}^* \leq 2^{s+1} \|v\|_{A^s} (2n)^{1/r}.
\]
The next step to extend the result to all odd occurrences is not clear. But once that is achieved, Proposition 3.2 is applied to show \( v \in \ell_{r,\infty} \) and is bounded above by \( A^s \) norm.

\[\Box\]

**Corollary 3.6** \(\dagger\). For \( 0 < r < 1 \) and \( \frac{1}{r} = s + 1 \), if \( v \in \ell_{r,\infty} \), then
\[
\|v - P_N v\|_1 \leq \left(\frac{1}{s} + 1\right) |v|_{r,\infty} N^{-s}.
\]

**Proof.** Fix \( v \in \ell_{r,\infty} \), the above conjecture states that \( v \in A^s \). Lemma 3.1 gives
\[
\|v - P_N v\|_1 \leq \|v\|_{A^s} N^{-s}.
\]
Applying the inequality (3.8) we get
\[
\|v - P_N v\|_1 \leq \left(\frac{1}{s} + 1\right) |v|_{r,\infty} N^{-s},
\]
where \( C > 0 \).

\[\dagger\]This result uses Conjecture \[3.2\] in the proof.
The support $N$ for the best $N$-term approximation solution of the CME is of the order $O(\varepsilon^{1/s})$. We set this as our benchmark. Solvers of the CME are required to produce supports of the same order or better to minimise memory size and increase speed-up. We have the framework required to define the Optimal Finite State projection method and prove that its support is of the same order as the best $N$-term approximation.

### 3.2 Optimal Finite State Projection method

The Optimal Finite State Projection method (OFSP) replaces the method of thresholding and hyper-rectangles by handling the support of the approximation. The aim is to have a method that is simple and gives a support closest to the minimal support. Being able to cheaply find the minimal support reopens the FSP to be able to compute a wide class of biological examples where it was previously considered to be too slow.

The OFSP gives structure to the support selection and can be utilised in all methods which are built upon the FSP framework. The Krylov FSP is such a method [8]. The Krylov FSP method has demonstrated quick computation time, however its domain size is the same as the FSP [8]. By applying the OFSP framework to the Krylov method, this method would compute over a smaller support size than the FSP.

The OFSP method is an addition of an best $N$-term approximation step to the FSP method. Adding this step and applying the best $N$-term approximation theory we derived above, we prove that the support of the OFSP is of optimal order. We start by stating the optimal order of an approximation, adapted from the work by Dahmen et. al. We then state the OFSP algorithm and prove that the generated approximation has optimal order support.

**Definition 3.7.** Let $f' \in \ell_1(\Omega)$ be an approximation of $f \in \ell_1(\Omega)$ with the condition that $\|f - f'\|_1 \leq \varepsilon$. Then the $\varepsilon$- approximation, $f'$, is said to have optimal order if

$$|\text{supp } f'| \leq C|f|_{r,\infty}^{1/s}\varepsilon^{-1/s},$$

where $C > 0$, $s = 1/r - 1$ and $r \in (0, 1)$.

This property ensures an optimal memory size for the approximation [11]. The current results in the literature do not indicate that FSP has optimal order...
support. The bounds

\[

\exp(A_J t) p_0 |J| \leq \exp(A_J t) p_0 |J| \leq \exp(A_J t) p_0 |J| + \varepsilon 1,
\]
given in Theorem 2.13 imply that any element which is \( \varepsilon \) or smaller can be truncated. However, this alone cannot guarantee the smallest support for that approximation. To attain optimality, we add an extra best \( N \)-term approximation step which ensures that the support of the approximation is small and is of optimal order. We describe the steps of the OFSP below.

**Algorithm 5: OFSP Algorithm**

\[
\text{Algorithm 5: OFSP Algorithm}
\]

\[
\begin{align*}
\text{input} & : p_0, \varepsilon, t \\
\text{output} & : \bar{p}_t \\
1 \text{ begin} & \\
2 & \hat{p}_t \leftarrow \text{FSP} \left( p_0, \varepsilon/2, t \right) \quad \text{(Algorithm 1)} \\
3 & \bar{p}_t \leftarrow \text{compress} \left( \hat{p}_t, \varepsilon/2 \right) \\
4 & \text{return } \bar{p}_t \\
5 \text{ end}
\end{align*}
\]

The procedure \text{compress} refers to the best \( N \)-term approximation of the vector. This procedure is computed as follows: Let \( p \) be the probability distribution and \( \varepsilon \) the compression amount. Then a compression is done by simply sorting \( p \) in descending order, and taking the \( N \) best terms such that the sum of the terms which are not selected sum up to \( \varepsilon \) or as close as possible. The terms that we keep give us a new approximation vector and new support.

By introducing the \text{compress} step, we are able to attain the optimal order shown in the following theorem.

**Theorem 3.8** (FKT Theorem 1). Let \( \bar{p}_t \in \ell_1(\Omega) \) be defined by the OFSP algorithm (Algorithm 3) with some \( \varepsilon > 0 \) and \( p_t \in \ell_1(\Omega) \) be the true solution. If \( \bar{p}_t \in \ell_{r,\infty} \) for some \( r \in (0, 1) \), then \( \bar{p}_t \) is optimal order approximation, that is,

\[
|\text{supp} \bar{p}_t| \leq C_1 |p_t|^{1/s} \varepsilon^{1/s}
\]

for some \( C_1 > 0 \), and \( \frac{1}{r} = s + 1 \).

**Proof.** Let \( p_t \in \ell_{r,\infty} \) for some \( r \in (0, 1) \).  

\( \dagger \)This result uses Conjecture [A] in the proof.
Let \( \hat{p}_t \) be the FSP approximation of \( p_t \) with error \( \varepsilon/2 \). Then, by the triangle inequality
\[
\|p_t - \bar{p}_t\|_1 \leq \|p_t - \hat{p}_t\|_1 + \|\hat{p}_t - \bar{p}_t\|_1 \leq \varepsilon.
\]
Furthermore Theorem 2.12 states that for all \( x \in \Omega \),
\[
0 \leq \hat{p}_t(x) \leq p_t(x).
\]
Applying Lemma 3.3 we have
\[
|\hat{p}_t|_{r,\infty} \leq |p_t|_{r,\infty}. \tag{3.9}
\]
As \( \bar{p}_t \) is the best \( N \)-term approximation of \( \hat{p}_t \) with error \( \varepsilon/2 \), Corollary 3.6 states
\[
|\text{supp } \bar{p}_t| \leq C|\hat{p}_t|_{r,\infty}^{1/s}(\varepsilon/2)^{-1/s},
\]
with
\[
\frac{1}{r} = s + 1 \text{ and } C > 0.
\]
Substituting (3.9) into the equation above gives us
\[
|\text{supp } \bar{p}_t| \leq C|p_t|_{r,\infty}^{1/s}(\varepsilon/2)^{-1/s}.
\]
Hence, by definition 3.7, OFSP has optimal order support. \( \square \)

We find that the OFSP method gives a significant speed up. Numerical experiments demonstrating the OFSP method are presented below.

### 3.3 Numerical Experiments

We now compare computation time, on two different systems, using the OFSP method, FSP method and the SSA. The OFSP and FSP methods are implemented using CMEPy \([4]\). The numerical experiments were run on a Quad-Core AMD Opteron(tm) Processor 8360 SE with 4GB of RAM.

The FSP implementation in CMEPy is different to the FSP algorithm given in Algorithm 1. Algorithm 1 is only a theoretical description some rearrangement needs to be considered for faster computation times. We give the algorithm for the implementation in Appendix A.

Applying the compression (c.f. Algorithm 5) at each time step is computationally expensive, especially when taking small time steps. Thus, the compression step was applied every few steps, the number of steps to compression for the particular problem is specified when the respective results are presented.
3.3.1 Goutsias’ Model

Goutsias’ model [38] describes the transcription regulation of a repressor protein in bacteriophage \( \lambda \) and involves six different species, \([M, D, RNA, DNA, DNA.D, DNA.2D]\), undergoing ten reactions.

<table>
<thead>
<tr>
<th>Reaction</th>
<th>Propensity</th>
<th>Rate Constant ( (s^{-1}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. RNA ( \xrightarrow{\kappa_1} ) RNA + M</td>
<td>( \alpha = \kappa_1[RNA] )</td>
<td>( \kappa_1 = 0.043 )</td>
</tr>
<tr>
<td>2. M ( \xrightarrow{\kappa_2} ) ( \emptyset )</td>
<td>( \alpha = \kappa_2[M] )</td>
<td>( \kappa_2 = 0.0007 )</td>
</tr>
<tr>
<td>3. DNA.D ( \xrightarrow{\kappa_3} ) RNA + DNA.D</td>
<td>( \alpha_3 = \kappa_3[DNA.D] )</td>
<td>( \kappa_3 = 0.0715 )</td>
</tr>
<tr>
<td>4. RNA ( \xrightarrow{\kappa_4} ) ( \emptyset )</td>
<td>( \alpha_4 = \kappa_4[RNA] )</td>
<td>( \kappa_4 = 0.0036 )</td>
</tr>
<tr>
<td>5. DNA + D ( \xrightarrow{\kappa_5} ) DNA.D</td>
<td>( \alpha_5 = \kappa_5[DNA][D] )</td>
<td>( \kappa_5 = 0.0019 )</td>
</tr>
<tr>
<td>6. DNA.D ( \xrightarrow{\kappa_6} ) DNA + D</td>
<td>( \alpha_6 = \kappa_6[DNA.D] )</td>
<td>( \kappa_6 = 0.4791 )</td>
</tr>
<tr>
<td>7. DNA.D + D ( \xrightarrow{\kappa_7} ) DNA.2D</td>
<td>( \alpha_7 = \kappa_7[DNA.D][D] )</td>
<td>( \kappa_7 = 10^{-5} )</td>
</tr>
<tr>
<td>8. DNA.2D ( \xrightarrow{\kappa_8} ) DNA.D + D</td>
<td>( \alpha_8 = \kappa_8[DNA.2D] )</td>
<td>( \kappa_8 = 0.8765 \times 10^{-11} )</td>
</tr>
<tr>
<td>9. M + M ( \xrightarrow{\kappa_{10}} ) D</td>
<td>( \alpha_9 = \kappa_9[M][M-1] )</td>
<td>( \kappa_9 = 0.008 )</td>
</tr>
<tr>
<td>10. D ( \xrightarrow{\kappa_{10}} ) M + M</td>
<td>( \alpha_{10} = \kappa_{10}[D][D-1] )</td>
<td>( \kappa_{10} = 0.5 )</td>
</tr>
</tbody>
</table>

SSA. We computed the Goutsias’ model with the initial condition, a point density at \((2, 6, 0, 2, 0, 0)\), propensities as given above and \( t_{final} = 300 \). The time partition for the OFSP and FSP methods (Appendix A) was set to \( \Delta t = 1 \), and the compression for the OFSP was conducted every 10 steps. To find the right number of realisations required for the SSA method for a global error \( 10^{-4} \), an FSP approximation of \( 10^{-6} \) as the reference solution, then 10,000 realisations at a time were computed until the difference between empirical distribution and the known distribution was less than \( 10^{-4} \). Approximately \( 10^6 \) runs were needed to find the right empirical distribution. In the table below we can see that the OFSP method is approximately 10 times faster than the FSP, the speed-up can be attributed to the OFSP only having 20% of the state space of the FSP method. Similarly the OFSP was significantly faster than the SSA, as the number of realisations needed to produce an empirical distribution with the right error are at least 40 times more than that of the number of state in the minimal support.

<table>
<thead>
<tr>
<th>( t_{final} = 300 )</th>
<th>OFSP ((h = 10))</th>
<th>FSP</th>
<th>SSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (Seconds)</td>
<td>116</td>
<td>1036</td>
<td>7475</td>
</tr>
<tr>
<td>Domain Size</td>
<td>22375</td>
<td>110931</td>
<td>( 10^6 ) runs</td>
</tr>
<tr>
<td>Global Error</td>
<td>( 10^{-4} )</td>
<td>( 10^{-4} )</td>
<td>( \approx 9.7 \times 10^{-3} )</td>
</tr>
</tbody>
</table>
3.3. NUMERICAL EXPERIMENTS

Figure 3.1: The Domain size at time step $t$, Goutsias’ model

Figure 3.2: Log Log plot of the Domain size at time step $t$, Goutsias’ model
Goutsias’ Model was chosen as it is considered to be really stiff, that is there are certain species which increase in population very quickly and others which change slowly, which makes domain selection difficult. Wolf et al have investigated this model with 500000 states using the SW method [88]. Similarly MacNamara et al computed the problem on a domain with 100000 states [59]. The results from MacNamara et al and Wolf et al took an order of hours to compute till $t_{final} = 300$. The delay in computation for MacNamara et al and Wolf et al is due to the computation of states which are not needed. In the computation above, the OFSP method identified that only 21000 states were needed to solve the problem up to a time horizon of 300. The OFSP support is only 4% of the support used by the Sliding Window. Hence, by implementing a compression step in their solver, the implementations by MacNamara et al and Wolf et al can be significantly faster.

In Figure 3.1, we show how the domain grows over time between the OFSP and the FSP. Appendix A presents the marginal distributions and their errors with respect to the SSA.

### 3.3.2 Goldbeter-Koshland Switch

The Goldbeter-Koshland switch [37] (G-K switch) consists of a pair of Michaelis-Menton enzyme kinetic models, catalysed by different enzymes, in which the product of one forms the substrate of the other, and vice versa. There are six chemical species $[S, E_1, C_1, P, E_2, C_2]$ undergoing six chemical reactions.

<table>
<thead>
<tr>
<th>reaction</th>
<th>propensity</th>
<th>rate constant ($s^{-1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 $A + E_1 \xrightarrow{\kappa_1} C_1$</td>
<td>$\alpha_1 = \kappa_1[S][E_1]$</td>
<td>$\kappa_1 = 1.0$</td>
</tr>
<tr>
<td>2 $C_1 \xrightarrow{\kappa_2} S + E_1$</td>
<td>$\alpha_2 = \kappa_2[C_1]$</td>
<td>$\kappa_2 = 1.0$</td>
</tr>
<tr>
<td>3 $C_1 \xrightarrow{\kappa_3} P + E_1$</td>
<td>$\alpha_3 = \kappa_3[C_1]$</td>
<td>$\kappa_3 = 0.1$</td>
</tr>
<tr>
<td>4 $P + E_2 \xrightarrow{\kappa_4} C_2$</td>
<td>$\alpha_4 = \kappa_4[P][E_2]$</td>
<td>$\kappa_4 = 1.0$</td>
</tr>
<tr>
<td>5 $C_2 \xrightarrow{\kappa_5} P + E_2$</td>
<td>$\alpha_5 = \kappa_5[C_2]$</td>
<td>$\kappa_5 = 1.0$</td>
</tr>
<tr>
<td>6 $C_2 \xrightarrow{\kappa_6} S + E_2$</td>
<td>$\alpha_6 = \kappa_6[C_2]$</td>
<td>$\kappa_6 = 0.5$</td>
</tr>
</tbody>
</table>

We compute the G-K switch using the FSP, OFSP and the SSA method for the initial condition, a point mass distribution at $(100, 100, 0, 0, 100, 0)$ until $t_{final} = 20$. The time step size $\Delta t = 0.001$ for the FSP and the OFSP, with the OFSP method compressing every 100 steps. It took approximately $10^6$ realisations for the empirical distribution from the SSA method to have a global error less than $10^{-4}$. What we see in this experiment is that the OFSP method is 50 times faster.
than the FSP and almost double that to SSA. We can see that the reason is that 2 percent of the support of the FSP approximation.

<table>
<thead>
<tr>
<th>$t_{\text{final}} = 20$</th>
<th>OFSP ($h = 100$)</th>
<th>FSP</th>
<th>SSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (Seconds)</td>
<td>214</td>
<td>10900</td>
<td>19844</td>
</tr>
<tr>
<td>Domain Size</td>
<td>4793</td>
<td>171006</td>
<td>$10^6$ Runs</td>
</tr>
<tr>
<td>Global Error</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
<td>$\approx 1.8 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

The Goldbeter-Koshland Switch is conventionally reduced to a three dimensional problem using the laws of conservation [37]. However, we compute the full problem to demonstrate the effectiveness of adaptive domain growth and compressions. The Goldbeter-Koshland is an example that Zhang et al [90] compute using multi-core architecture. The problem is solved over a full grid of 700000 states [90]. The computation times over 16 cores by Zhang et al are approximately similar to the single core of the OFSP. This is because the OFSP only solves close to the minimal support and for this problem only needed 5000 states. By replacing the FSP with the OFSP; the computation time of the implementation by Zhang et al can be significantly faster than the CMEPy implementation.

In Figure 3.3, we show how the domain grows over time between the OFSP and the FSP. Appendix A presents the marginal distributions and their errors with respect to the SSA.

3.4 Discussion

In this chapter, we introduced the Optimal Finite State Projection method and proved that its support is of the same order as the support of the best $N$-term approximation. Upon implementation, the OFSP method performs faster than the conventional FSP method. The truncation of the state space every few steps ensures that the support is close to the minimal support, which then gives a smaller problem to compute. This implies a computational speed up. The log-log plots of the domain size versus the time was provided to show that the support grows polynomially. This is a key indicator that the domain selection for solving the CME should be adaptive.

In this chapter, a translation of the famous result by Devore et. al. [17] is given for the $\ell_1$ norm. It is shown that achieving this result for the $\ell_1$ norm has its obstacles. It is tempting to take the $\ell_2$ norm, so that the results by Devore et. al. can be applied directly. To use the $\ell_1$ or the $\ell_2$ norm is still open in the CME community as each has its benefits. The $\ell_2$ norm is a popular error norm and has
CHAPTER 3. OPTIMAL FINITE STATE PROJECTION METHOD

Figure 3.3: The Domain size at time step $t$, G-K Switch

Figure 3.4: Log Log plot of the Domain size at time step $t$, G-K Switch
3.4. DISCUSSION

an inner product structure which gives many strong theoretical results. However, in the CME case, it does not give as strong results as the $\ell_1$ norm. That is, with the $\ell_1$, the CME operator is invariant over time. Furthermore, the $\ell_1$ norm gives a point-wise error ($\ell_\infty$ norm). This is an important feature which allows the building of an approximation constructively. This feature is missing the $\ell_2$ case. While the $\ell_2$ the approximation gets better, we cannot get an estimate of the $\ell_\infty$ norm. For example, we might add basis which improves the $\ell_2$ norm, however this does not imply that the $\ell_{r,\infty}$ norm has improved. Furthermore, the CME operator with respect to the $\ell_2$ norm is not invariant. The solution of the CME gets wider and spreads. Thus, the $\ell_2$ norm decreases over time. The positive note of the $\ell_2$ norm is the ability to apply strong theoretical results to the problem, which are not so evident under the $\ell_1$ norm.

The OFSP method indicates that the support is order minimal after completing the computation. While this in itself has resulted in a significant speed-up, we are left with the question: Can we find the optimal order domain for the next time step before completing the time step? We address this question in Section 5. In the following section, we introduce an algorithm to parallelise the OFSP method to utilise HPC clusters.
Chapter 4

The Parallel Finite State Projection Method (PFSP)

The Parallel Finite State Projection method utilises the linearity of the CME problem to distribute computation over multiple cores. The need for parallelising the CME arises when we compute solutions of complex biological systems with multiple interacting species. Currently, stochastic simulation methods are used to compute CME approximations for such problems due to the large and intricate state spaces, making them difficult to solve directly. Stochastic simulation methods compute independent realisations and average over them to construct the approximation of the underlying probability distribution. Stochastic simulation methods have been successful since each realisation can be run in parallel and no knowledge of the domain is required as it is realised progressively. Since the realisations are independent, we get a linear or near linear speed up when we distribute the problem over multi-core architectures. Even though the stochastic simulation methods are simple and bypass the problem of finding the domain, which the direct solver struggles with, the methods have to compute many realisations to construct an accurate approximation. That is, they are required to compute more realisations than the minimal support of a direct method. This result is discussed further in Section 4.2.

There are two methods to approximate the solution of the CME directly using multi-core architecture. The first method is the Parareal method, proposed by Engblom in 2010 in the CME context [26] and the second is an FSP method implemented [90] by Zhang in 2009. Both these methods showed merit in utilising multiple cores to compute an approximation for the CME.

The Parareal method partitions the problem in time rather than in state space
That is, on each core we compute a different time interval but over the same state space. We iteratively construct our solution in each time interval on each core. The previous iteration is shared globally to help refine the next iteration on each core. This method is explained in further detail in Section 4.3. The Parareal method was proposed by Engblom to help compute the realisations of RDME problems [26]. However, the Parareal method cannot be extended to directly compute Reaction Diffusion Master Equation (RDME) or large CME problems due to being unable to distribute the domain. In the Parareal method we compute all time steps at once. This puts a constraint on the state space to be the union of all the minimal state spaces over all time up to the final time step. With an RDME type problem, the state space would be in the order of millions or billions on each core, making the method too difficult to compute on existing hardware.

Earlier to Engblom, Zhang proposed a straightforward method to use an FSP method with domain decomposition [90]. Zhang proposed the state space to be partitioned into sub-domains, where each core solves the CME on a different sub-domain. As the CME is hyperbolic in nature, the probability will flow from one sub-domain to another over time. The cores would have to buffer the probability flowing out of the sub-domain and communicate the information over to the adjacent cores. The domain decomposition aspect of this method helps give a speed up for a small number of cores [90]. However, the partitions being fixed through the computation give rise to an increased amount of communications between the cores and large domain selections. This in turn adds delay to the computation time and forces hyper rectangular domains which are computationally infeasible, as discussed in Chapter 3.

In summary, methods proposed by Engblom and Zhang ported the CME problem well onto the multi-core framework, however they are not recommended over a stochastic simulation method to compute RDME or large CME problems. Large CME or RDME models have an inherently large minimal support size, hence domain selection and construction are important aspects of being able to directly solve the CME. We collate the positives and negatives in the methods mentioned above and note that to directly compute the solution of CME/RDME on multi-core architectures, the solver we propose must have the following features:

- Order minimal supports for approximations grow the domain adaptively, keeping it close to the optimal.
4.1 Reaction-Diffusion Example

The Reaction-Diffusion Master Equation (RDME) is an extension to the CME. When considering a large physical space in which particles are undergoing re-

- Domain partitioning to partition and distribution onto multi-core.
- At most linear amount of communications between cores, so we can up scale and have the ability to use many cores.

In Chapter 3 we see that the OFSP method gives an approximation whose support is order optimal. That is, it is of the order of the minimal support. We see that by constructing the domain ongoingly, the support of our approximation stays close to the minimal support size and decreases our computation time.

In late 2010, Sunkara and Hegland proposed extending the OFSP method onto a multi-core architecture called the Parallel Finite State Projection method (PFSP method) [82]. The PFSP method as proposed by Sunkara and Hegland uses the linearity of the CME problem to distribute the domain onto multiple cores and compute an OFSP on each core, hence keeping the support of the approximation of each core to the optimal order. Instead of communicating the information on the boundary, the PFSP method grows its boundary on each core as needed and collates the information to sum up the overlaps. This is the only communication between cores in the PFSP method. Having to compute multiple copies of the same state is the overhead of the PFSP method. However, the PFSP method does give a tighter approximation, that is, the approximation has optimal order support, which is a crucial in large domain problems. This theorem and others are given in Section 4.4.2. The PFSP method was constructed to successfully incorporate the features mentioned above in the algorithm to get a speed up over a stochastic simulation method for RDME problems. Sunkara and Hegland state the algorithm and theorems demonstrating the methods accuracy, however no empirical experiments were conducted.

In this chapter we recall the PFSP method proposed by Sunkara and Hegland and empirically compare the method to the SSA and the Parareal method. We start with a notational introduction to the RDME problems followed by a brief discussion of stochastic simulation methods and the Parareal method. We then state the PFSP method and derive bounds regarding the error and support. The chapter concludes with numerical experiments to compare the SSA, Parareal and the PFSP method and discusses further research.
actions, the transitional probability assumptions no longer hold. That is, the particles are too far from each other and the transitional probability is no longer an accurate description of the system transitions. The physical space that the system is in is so large that the diffusion of particles in the physical space also has to be considered. The reaction-diffusion master equation discretises the physical domain into voxels (compartments). The voxel size is chosen to ensure that the particles are well mixed and in thermal equilibrium inside each voxel. In doing this, the transition probabilities of the CME are then valid again inside each individual voxel. Thus, the probability distribution of such a system is simply given by the CME describing reactions inside each voxel plus another CME describing the diffusions between voxels.

We formulate the RDME as follows. Let us assume that inside each voxel, there are $N_s \in \mathbb{N}$ different species and they are undergoing $N_r \in \mathbb{N}$ reactions. The number of voxels in the physical domain is denoted as $N_v \in \mathbb{N}$.

We denote the populations of the $N_s$ species in the $i$th voxel as $x_i \in \mathbb{N}_0^{N_s}$. Then a state which the reaction-diffusion system can be in at any time is denoted by $x = (x_i)_{i=1}^{N_v}$. We let $\Omega$ denote all the population configurations, $x$, the reaction-diffusion system can undertake through all time.

Let $v_j$ denote the stoichiometric vector and $\alpha_j$ the propensity function for the $j$th reaction in a voxel. The propensity and stochiometry inside a voxel is as in the formulation for the CME. We define $v_{i,j} := (\delta_i(k)v_j)_{k=1}^{N_v}$ as the stoichiometric vector of the $j$th reaction in the $i$th voxel. The $\delta$ is a Kronecker delta.

The reaction contribution for the probability of the system being in a population configuration $x$ at time $t > 0$ is given by,

$$\frac{\partial P(x; t)}{\partial t} = \sum_{i=1}^{N_v} \sum_{j=1}^{N_r} \alpha_j(x_i - v_j)P(x - v_{i,j}; t) - \alpha_j(x_i)P(x; t).$$

Considering the above equation over $x \in \Omega$, we get the CME type matrix representation,

$$\frac{dp_t}{dt} = ARp_t, \quad (4.1)$$

with $p_t$ denoting a vector with values $P(x; t)$ indexed by $x \in \Omega$.

Let $N_d > 0$ denote the number of different diffusions that can occur out of each voxel. We define the $\beta_k$, for $k = 1, \ldots, N_d$, to be the propensity function of the $k$th type of diffusion occurring in a voxel. The propensity of the diffusions have
4.2. SSA PARALLELISATION

similar property to the propensities of the reactions, that is, for \( k = 1, \ldots, N_d \)

\[
\beta_k(x) := \begin{cases} 
> 0 & \text{if } x \in \Omega, \\
0 & \text{otherwise}. 
\end{cases}
\]

Let \( d_{i,k} \) for \( k = 1, \ldots, N_d \) and \( i = 1, \ldots, N_v \) denote the stoichiometric vectors which describes the change in population configuration if \( k \)th kind of diffusion occurred in the \( i \)th voxel.

The diffusion contribution to \( P(x; t) \) is given by,

\[
\frac{\partial P(x; t)}{\partial t} = \sum_{i=1}^{N_v} \sum_{k=1}^{N_d} \left( \sum_{x-d_{i,k} \geq 0} \beta_k(x - d_{i,k}) P(x - d_{i,k}; t) - \sum_{x+d_{i,k} \geq 0} \beta_k(x) P(x; t) \right).
\]

Considering the above equation over all \( x \in \Omega \), we get the matrix formulation,

\[
\frac{dp_t}{dt} = A_D p_t, \tag{4.2}
\]

with \( p_t \) denoting a vector with values \( P(x; t) \) indexed by \( x \in \Omega \).

Combining the diffusion contribution (4.2) and the reactions contribution (4.1), the formulation for the probability distribution over \( \Omega \) is given by

\[
\frac{dp_t}{dt} = (A_R + A_D) p_t.
\]

The equation above is referred to as the Reaction-Diffusion Master Equation.

The increase in voxels increases the dimension linearly. Hence RDME systems have very large number of particles. Due to the really high dimension, the stochastic simulation methods are the method of choice in the literature \[89, 79, 28, 30\]. In section 4.5 we use the Parallel Finite State Projection method to construct the probability distribution for a simple 20 dimensional reaction-diffusion model.

4.2 SSA Parallelisation

The stochastic simulation methods have been the most successful to compute large CME and RDME type models \[19, 23, 89, 75\]. Stochastic simulation methods are easy to implement and the speed up expected is nearly proportional to the number of cores we allocate. Furthermore we realise the domain ongoingly rather than having to estimate it a priori, which is the major issue for CME problems. The SSA is the framework on which hybrid methods have been built.
upon. Lötstedt et. al and Khammash et. al have proposed extensions to the SSA to help faster solving of a specific class of problems in the RDME models [30, 21].

In this chapter we only consider the SSA method, that is, we are investigating the support size generated by realisation based approximations. The hybrid and advanced methods which superseded the SSA help compute each realisation faster, however they don’t affect the support size. Hence, we study the SSA and the results that arise can be extended to the superior methods [34, 3, 71, 83, 10, 62]. A review of the SSA method is found in Section 2.4.

To compare the realisation based solvers with the direct solvers for the CME, consider the following question.

Given an error tolerance, what is the minimum number of realisations needed to construct an approximation with the global error, in $\ell_1$ norm, below tolerance?

This section aims to link the minimal realisations to the minimal support. When approximating a probability distribution, the bottleneck for an FSP type method, is in finding the minimal support and the bottleneck for an SSA method is the number realisations required. At a glance these two factors are different quantities, however these two quantities are good indicators for deciding on which method to use. We proceed by introducing notation and then present the theorem linking the minimal realisation with the minimal support.

We start by considering the class of approximations that arise from stochastic simulations with $n$ realisations. Let $u \in \ell_1$ be the probability distribution we are approximating. We define for $n \in \mathbb{N}$,

$$\chi_n := \left\{ w \in \ell_1 : \text{for } i = 1, 2, \ldots, w_i = \frac{k_i}{n} \text{ with } k_i \in \{0, 1, \ldots, n\} \text{ and } \|w\|_1 = \sum_{i=1}^{\infty} \frac{k_i}{n} = 1 \right\}.$$ 

The set $\chi_n$ is the set of all empirical distributions, which can be constructed via a stochastic simulation method with $n$ realisations. We denote a projection $P_\theta$ to be the best $N$-term approximation of error $\theta > 0$. That is for $u \in \ell_1$ we have the result $\|u - P_\theta u\|_1 = \theta$. The best $N$-term approximation is given in detail in section 3.1.

**Theorem 4.1.** Let $u \in \ell_1$ be a probability distribution and $\theta > 0$. For $n \in \mathbb{N}$ if there exists $w \in \chi_n$ such that

$$\|w - u\|_1 \leq \theta,$$

then

$$\text{supp } P_\theta u \leq n.$$ 

$P_\theta$ denotes the best $N$-term approximation of $u$ of error $\theta$. 

4.2. SSA PARALLELISATION

Proof. We shall assume that $n < \text{supp} P_\theta u$ and show we have a contradiction.

Let $N := \text{supp} P_\theta u$ and assume that for some $m < N$, there exists $w \in \chi_m$ such that

$$\|w - u\|_1 \leq \theta. \quad (4.3)$$

We recall from Section 3.1.1 the space $\Sigma_N$, which is the space all sequences with support size up to $N$. We also recall for $P_\theta$, the best $N$-term approximation operator, that

$$\|P_\theta u - u\|_1 := \min_{v \in \Sigma_N} \|u - v\|_1.$$

Since $w \in \chi_m$ and $m < N$, we have $w \in \Sigma_N$, as its support is less than $N$. By applying the definition of $\Sigma_N$, we get

$$\|P_\theta u - u\|_1 < \|u - w\|_1, \text{ for } w \neq P_\theta u. \quad (4.4)$$

By definition, $\|P_\theta u - u\|_1 = \theta$. We know the support of $w$ is smaller than $u$, hence (4.4) holds.

$$\theta = \|P_\theta u - u\|_1 < \|u - w\|_1.$$

The equation above contradicts with (4.3).

Therefore to find a $w \in \chi_m$ such that $\|w - u\|_1 \leq \theta$, we need $m \geq N$. \qed

Remark 4.2. A single realisation of an SSA is very cheap; it has a memory complexity of $N_s$. When using realisations to construct an empirical distribution, it is known that for an approximation error $\varepsilon > 0$, $C\varepsilon^{-2}$ realisations are needed, where $C > 0$. Using Theorem 4.1, we can show that $C$ is bounded below by the number of realisations multiplied by $\varepsilon^2$. We know that the support of the CME scales $\sim d^{N_s}$, where $d > 1$. By Theorem 4.1 we can conclude that $C \geq d^{N_s}\varepsilon^2$.

The theorem above states, given an error $\theta > 0$, the cardinality of the minimal support of error $\theta$ will always be less then or equal to the number of realisations required to generate an empirical distribution of the same error. For high dimensional problems, the stochastic simulation is used with the assumption that it is a simpler problem to compute. However, Theorem 4.1 indicates to us that if we knew the minimal support, then solving the CME would have a smaller order of complexity to computing realisations. In Theorem 4.1 we do not indicate the magnitude of the difference between the two methods. Hence, in section 4.5 we compare the number of realisations with the size of the minimal support to approximate a Poisson distribution.
It was shown by many authors that direct solvers are faster to compute than stochastic simulation methods \[8, 42, 88, 46\]. Burrage et. al showed that particular problems up to 22 dimensions could be computed faster than the SSA method. However, it was an open question on how large a dimension will the direct method be faster and at what point should we shift over to a stochastic simulation approximation. Theorem 4.1 infers that for CME problems where we know the minimal support, the direct method would have fewer computations than an SSA, theoretically making it faster. This ties in with the discussion on the importance of minimal support in Section 3.2.

Even though stochastic simulation methods require more realisations, the difficulty in computation for finding the minimal support and avoiding linear computation makes stochastic simulation methods more favourable to implement. The following is the multi-core SSA algorithm.

**Algorithm 6: Parallel SSA**

```
input : N, K, X₀, t₀, t
output: wₜ
begin
1      \( N_k \leftarrow N/K \)
2      for core \( k \in 1, \ldots, K \) do
3         \( w^kₜ \leftarrow \text{SSA}(X₀, N_k, t₀, t) \)
4      end
5      \( wₜ \leftarrow \text{concatenate} \{w^kₜ : k = 1, \ldots, K\} \)
7 end
```

Any realisation based method can be inserted into step 4. In Section 4.5 we analyse and compare implementations of the Parallel SSA for RDME problems. We now consider the Parareal method for distributing the CME problem onto multi-core architecture.

### 4.3 Parareal Method

The Parareal method helps parallelise the CME problem with respect to time. Each core computes a different time interval. The Parareal method is used for computing smaller problems for large time range. We give the construction and algorithm of the Parareal method below.

We recall the CME problem is given as follows.
\[ \frac{dp_t}{dt} = Ap_t, \quad (4.5) \]

where \( A \in \mathbb{R}^{n \times n} \), \( p_t \in \mathbb{R}^n_+ \). The initial value \( p_0 \in \mathbb{R}^n_+ \), has the condition \( \|p_0\|_1 \leq 1 \).

Here \( A \) is the generator matrix which has the properties:

- the diagonal elements of \( A \) are negative,
- the off diagonal elements are all positive,
- the columns of \( A \) sum up to zero.

The strategy is to decompose the time interval uniformly to distribute onto different cores. Let \( K \) be the number of cores. We pick \( K \) equidistant points in \((0, T)\) to get

\[ 0 < T_1 < T_2 < \ldots < T_K = T, \]

where differences are \( \Delta T \). For each compartment we have the problem

\[
\begin{align*}
\frac{dy_k}{dt} &= Ay_k, \text{ for } t \in [T_k, T_{k+1}] \\
y_{T_k}^k &= y_{T_k}^{k-1}, y_0 = p_0
\end{align*}
\quad (4.6)
\]

for \( k = 0, \ldots, K - 1 \). For consistency with other parallel methods discussed in this chapter, the superscript index is an indicator for component rather than a power.

The collection of solutions of (4.6), denoted by \( \{v_0, v_1, \ldots, v_{K-1}\} \), are connected to the solution of (4.5), \( p \), if and only if for any \( k = 0, \ldots, K - 1 \),

\[ v^k = p_{T_k}. \]

Or when written with respect to (4.6),

\[ v_k = y_{T_k}^{k-1} \text{ with } v_0^{-1} = p_0. \]

Let \( \mathcal{F}_{\Delta t} \) be a fine propagator such that for any given \( \mu \), \( \mathcal{F}_{\Delta t}(\mu) \) is the solution at \( \Delta T \) of (4.5).

We write (4.6) in the matrix form as follows.

\[
\begin{pmatrix}
I & 0 & 0 & \ldots & 0 \\
0 & -\mathcal{F}_{\Delta t} & I & 0 & \ldots \\
\vdots & \ddots & \ddots & \ddots & \ldots \\
0 & \ldots & 0 & -\mathcal{F}_{\Delta t} & I
\end{pmatrix}
\begin{pmatrix}
v^0 \\
v^1 \\
\vdots \\
v^{K-1}
\end{pmatrix}
= \begin{pmatrix}
p_0 \\
v_0 \\
\vdots \\
0
\end{pmatrix}
\quad (4.7)
\]
The system above is the Gauss-Seidel waveform relaxation of a Linear ODE. Hence, to solve for \( v \), we can draw upon the result that if the derivative of \( F_{\Delta t} \) is Lipschitz in the \( \ell_1 \) norm, we can construct a sequence \( v^n \) which converges to \( v \). That is,

\[
\| v^n - v \|_1 \leq (CT)^n \| v^0 - v \|_1,
\]

where \( C \) is the Lipschitz constant and \( T \) is the time interval length. Staff proposed the Parareal method in 2003, a particular selection of the sequence \( v^n \). Staff proposed to approximate the inverse of \( F_{\Delta t} \) by using a coarse operator, \( G_{\Delta t} \), and an implicit Euler scheme. That is,

That is, to use the predictor-corrector scheme defined as

\[
\begin{cases}
v^{k,n} := F_{\Delta T}(v^{k-1,n-1}) + G_{\Delta T}(v^{k-1,n}) - G_{\Delta T}(v^{k-1,n-1}), \\
v^{0,0} = p_0,
\end{cases}
\]

for \( n \in \mathbb{N} \). Here \( v^{k,n} \) is the \( n \)th iteration of the \( k \)th component of \( v^n \). A survey detailing the steps is found in [77, 65]. We see that \( F_{\Delta T}(v^{k-1,n-1}) \) can be computed in parallel and \( G_{\Delta T}(v^{k-1,n}) \) is computed in serial over \( k \).

We let \( \tau > 0 \) with

\[
\| v^{n+1} - v^n \|_2 \leq \tau,
\]

be the stopping criterion for the iteration. There is an optimal number of iterations after which the residuals do not change. It is shown that the selection of the coarse operator, given it is stable and convergent for the time step size, does not affect the accuracy. However, the operator selection does affect the rate of convergence of the iterations.
4.4 PFSP Method

To successfully compute an FSP type method for large CME or RDME type problems, we need the following three key features,

- distribution of domain onto multiple cores,
- minimal support size on each core,
CHAPTER 4. THE PARALLEL FINITE STATE PROJECTION METHOD (PFSP)

• at most linear to the number of cores of communication.

The PFSP method addresses two of these three features in its structure. The PFSP method partitions the domain and computes an OFSP on each of the partitions in parallel. To compute an OFSP on each core helps grow the partition on demand. This feature aids in computation of large dimensional problems where considering hyper-rectangular domain approximations are not feasible. The linearity of the CME problem allows us to prove the global error is sub-additive in the local errors on each core, shown in Theorem 4.4.2. Furthermore the approximation is enveloped between zero and the real solution, as is the case in the FSP method (Theorem 2.12). The linearity of the CME problem can also be used to show that the support of the PFSP method has optimal order, Theorem 4.6, therefore minimising our computation. We begin by introducing the PFSP notation, followed by stating the PFSP algorithm. We then prove the error bounds and support bounds for the PFSP method.

4.4.1 PFSP Notation

The following notation is presented to describe parts of the CME problem on different cores. We use the notation superscript $k$ when we are referring to a vector or state spaces on the $k$th core, and subscript $k$ when referring to the local errors on the $k$th core. We wish to distribute the state space of the CME problem onto multiple cores. Hence let $Z$ be a partitioning operator with the following properties:

$$Z : \Omega \mapsto \{\Omega^k : k = 1, \ldots, K\}$$

where

$$\Omega^k \cap \Omega^j = \emptyset$$

if $k \neq j$ and $\Omega^k \subset \Omega$ for all $k$. Furthermore, $\Omega = \bigcup_{k=1}^{K} \Omega^k$.

For $k = 1, \ldots, K$ define $u^k_t : \Omega \to [0, 1]$ by,

$$(u^k_t)(x) := \begin{cases} (p_t)(x) & \text{if } x \in \Omega^k \\ 0 & \text{Otherwise.} \end{cases} \quad (4.9)$$

Then,

$$p_t = \sum_{k=1}^{K} u^k_t. \quad (4.10)$$
4.4. **PFSP METHOD**

The linearity of the CME problem (§2.1.1) gives the following equality,

\[ p_{t+\Delta t} = e^{A\Delta t} p_t \]  
\[ = e^{A\Delta t} \sum_{k=1}^{K} u_{t}^{k} \]  
\[ = \sum_{k=1}^{K} e^{A\Delta t} u_{t}^{k}. \]

We can thus evolve \( p_t \) by evolving the components \( u_t^{k} \) in parallel.

The Parallel Finite State Projection method is described below in Algorithm 8 and Algorithm 9. The PFSP algorithm partitions the domain into non-intersecting sub-domains and computes the CME of each sub-domain independently, for a given time step, on different cores. Then, the probability over the individual sub-domains is collated and the new domain is repartitioned to compute the distribution at the next time step.

We give the input parameters for the PFSP method.

<table>
<thead>
<tr>
<th>Table 4.1: PFSP input parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_0 )</td>
</tr>
<tr>
<td>( K \in \mathbb{N} )</td>
</tr>
<tr>
<td>( t &gt; 0 )</td>
</tr>
<tr>
<td>( \varepsilon &gt; 0 )</td>
</tr>
<tr>
<td>( \varepsilon_k &gt; 0, \text{ for } k = 1, \ldots, K )</td>
</tr>
<tr>
<td>( \Delta t &gt; 0 )</td>
</tr>
<tr>
<td>( h \in \mathbb{N} )</td>
</tr>
<tr>
<td>( \varepsilon_c &gt; 0 )</td>
</tr>
</tbody>
</table>

The PFSP algorithm (Algorithm 8) starts by computing an OFSP in serial until time \( t > 0 \). This serial step is introduced to ensure that there are at least \( K \) distinct states in the support, so that a proper partition is achieved. If there are more than \( K \) points in the initial distribution, that initial OFSP step can be omitted. The OFSP step outputs an approximation for time \( t \). We then partition the support of the approximation into \( K \) partitions (projection). Each partition and their respective state probabilities, are sent to a core to compute the CME for a time step of \( \Delta t \) and error \( \varepsilon_k \), where \( \varepsilon_k \) is the approximation error allowed on the \( k \)th core. Once the approximation on each core is constructed, the states and their probabilities are collated. Using the linearity of the CME
Algorithm 8: PFSP Master Algorithm

input: \( p_0, K, \varepsilon, \varepsilon_1, \ldots, \varepsilon_K, \varepsilon_c, t, \Delta t, h \)
output: \( \bar{u}_{t+h\Delta t} \)

1 begin
2 \( \bar{u}_t, \bar{\Omega}_t \leftarrow \text{OFSP}(p_0, \varepsilon, t) \)
3 \( s \leftarrow t \).
4 for \( i \leftarrow 1 \) to \( h \) do
5 \( \bar{u}_1^s, \bar{u}_2^s, \ldots, \bar{u}_k^s \leftarrow \text{projection}(\bar{u}_s, Z(\bar{\Omega}_s, K)) \)
6 for \( k \leftarrow 1 \) to \( K \) do
7 send \( (\bar{u}_k^s, \varepsilon_k, \Delta t) \) to Core \( k \)
8 \( \hat{u}_s^{k+\Delta t} \leftarrow \text{receive from Core } k \)
9 end
10 \( \bar{u}_{s+\Delta t} \leftarrow \sum_{k=1}^{K} \hat{u}_k^{s+\Delta t} \)
11 \( \bar{u}_{s+\Delta t}, \bar{\Omega}_{s+\Delta t} \leftarrow \text{compress}(\bar{u}_{s+\Delta t}, \varepsilon_c) \)
12 \( s \leftarrow s + \Delta t \)
13 end
14 return \( \bar{u}_{t+h\Delta t} \)
15 end

Problem, probabilities over the same states are summed. Then a compression step (compress) is taken, that is, we throw away the largest subset of the support which has a cumulative probability of \( \varepsilon_c \). This compression step removes redundant states. The compressed support and the distribution are set as initial conditions and evolved another time step \( \Delta t \) via the steps mentioned above. In section 4.4.2, we describe how the selection of the error parameters gives bounds on the global error.

Remark 4.3. Algorithm 9, PFSP Slave Algorithm, is stated as an OFSP method.

Algorithm 9: PFSP Slave Algorithm

input: \( \hat{u}_t^k, \varepsilon_k, \Delta t \)
output: \( \hat{u}_t^k \)

1 begin
2 return \( \hat{u}_t^k \leftarrow \text{OFSP}(\hat{u}_t^k, \varepsilon_k, \Delta t) ; \)
3 end
However this is not necessary as this part of the solver can be replaced by any other methods for solving the CME, for example, the Krylov FSP \cite{8}, Wavelet method, \cite{16}, Spectral approximation \cite{27}, QSSA \cite{59}, Sparse Grids \cite{42} or even the SSA.

We show the accuracy of $\tilde{u}_{t+h\Delta t}$ generated by Algorithm 8 and also that the support size of this approximation has optimal order.

### 4.4.2 Error and Support Bounds

We give an error bound of the approximation generated by the PFSP method, Algorithm 8, for a single parallelised step ($h = 1$), then the error bound for $h > 1$ follows. We use best $N$-term approximation results, derived in Chapter 3, to prove that the support of the approximation from the PFSP method has optimal order.

**Theorem 4.4.** Let $\tilde{u}_{t+\Delta t}$ be the PFSP approximation, Algorithm 8, of the CME problem (2.1.1), with input parameters as given in Table 4.1. Let $p_t, p_{t+\Delta t}$ be the solutions of the CME problem at $t$ and $t + \Delta t$ respectively. If $h = 1$, then

$$\|p_{t+\Delta t} - \tilde{u}_{t+\Delta t}\|_1 \leq \sum_{k=1}^K \varepsilon_k + \varepsilon + \varepsilon_c. \quad (4.14)$$

If $h > 1$, then

$$\|p_{t+h\Delta t} - \tilde{u}_{t+h\Delta t}\|_1 \leq h \left( \sum_{k=1}^K \varepsilon_k + \varepsilon_c \right) + \varepsilon. \quad (4.15)$$

**Proof.** We start with,

$$p_{t+\Delta t} - \tilde{u}_{t+\Delta t} = p_{t+\Delta t} - \tilde{u}_{t+\Delta t} + \tilde{u}_{t+\Delta t} - \tilde{u}_{t+\Delta t}.$$

Applying the triangle inequality gives,

$$\|p_{t+\Delta t} - \tilde{u}_{t+\Delta t}\|_1 \leq \|\tilde{u}_{t+\Delta t} - \tilde{u}_{t+\Delta t}\|_1 + \|p_{t+\Delta t} - \tilde{u}_{t+\Delta t}\|_1.$$

By construction (Algorithm 8) $\tilde{u}_{t+\Delta t}$ is an approximation of $p_{t+\Delta t}$ of error $\varepsilon + \sum_{k=1}^K \varepsilon_k$. In Algorithm 8 we choose $\|\tilde{u}_{t+\Delta t} - \tilde{u}_{t+\Delta t}\|_1$ to have error $\varepsilon_c$. Combining we get,
CHAPTER 4. THE PARALLEL FINITE STATE PROJECTION METHOD (PFSP)

\[ \| p_{t+\Delta t} - \bar{u}_{t+\Delta t} \|_1 \leq \sum_{k=1}^{K} \varepsilon_k + \varepsilon + \varepsilon_c. \]

Fix \( h > 1 \). Let \( A^{k,h} \) be the sub-matrix of \( A \) on the \( k \)th core and \( h \)th step. Furthermore, we denote \( P_N \) as the best \( N \)-term approximation operator (c.f. §3.1) or error \( \varepsilon_c \).

Let \( A_{k,h} \) be the sub-matrix of \( A \) on the \( k \)th core and \( h \)th step.

Furthermore, we denote \( P_N \) as the best \( N \)-term approximation operator (c.f. §3.1) or error \( \varepsilon_c \).

\[
\left\| e^{A_{t+(h-1)\Delta t}} \bar{u}_{t+(h-1)\Delta t} - \hat{u}_{t+h\Delta t} \right\|_1 = \left\| e^{A_{t+(h-1)\Delta t}} \bar{u}_{t+(h-1)\Delta t} - \sum_{k=1}^{K} P_N e^{A^{k,h}_{t+(h-1)\Delta t}} \bar{u}_{t+(h-1)\Delta t} \right\|_1,
\]
\[
= \sum_{k=1}^{K} \left\| e^{A_{t+(h-1)\Delta t}} \bar{u}_{t+(h-1)\Delta t} - P_N e^{A^{k,h}_{t+(h-1)\Delta t}} \bar{u}_{t+(h-1)\Delta t} \right\|_1,
\]
\[
\leq \sum_{k=1}^{K} \left\| e^{A_{t+(h-1)\Delta t}} \bar{u}_{t+(h-1)\Delta t} - P_N e^{A^{k,h}_{t+(h-1)\Delta t}} \bar{u}_{t+(h-1)\Delta t} \right\|_1,
\]
\[
\leq \sum_{k=1}^{K} \varepsilon_k. \quad (4.16)
\]

We construct a recursion to give our error bound. We have

\[
\| p_{t+h\Delta t} - \hat{u}_{t+h\Delta t} \|_1 \leq \| p_{t+h\Delta t} - e^{A_{t+(h-1)\Delta t}} \bar{u}_{t+(h-1)\Delta t} \|_1 + \| e^{A_{t+(h-1)\Delta t}} \bar{u}_{t+(h-1)\Delta t} - \hat{u}_{t+h\Delta t} \|_1,
\]
\[
\leq \| e^{A_{t+(h-1)\Delta t}} p_{t+(h-1)\Delta t} - e^{A_{t+(h-1)\Delta t}} \bar{u}_{t+(h-1)\Delta t} \|_1 + \sum_{k=1}^{K} \varepsilon_k, \quad \text{by applying (4.16),}
\]
\[
\leq \| p_{t+(h-1)\Delta t} - \bar{u}_{t+(h-1)\Delta t} \|_1 + \sum_{k=1}^{K} \varepsilon_k, \quad \text{Theorem 2.11} \quad (4.17)
\]

At the compression step, we get the inequality,

\[
\| p_{t+h\Delta t} - \bar{u}_{t+h\Delta t} \|_1 \leq \| p_{t+h\Delta t} - \hat{u}_{t+h\Delta t} \|_1 + \| \hat{u}_{t+h\Delta t} - \bar{u}_{t+h\Delta t} \|_1,
\]
\[
\leq \| p_{t+h\Delta t} - \hat{u}_{t+h\Delta t} \|_1 + \varepsilon_c,
\]
\[
\leq \| p_{t+(h-1)\Delta t} - \bar{u}_{t+(h-1)\Delta t} \|_1 + \sum_{k=1}^{K} \varepsilon_k + \varepsilon_c, \quad \text{by applying (4.17).}
\]

By expanding the right-hand size recursively, we get
\[ \| p_{t+h\Delta t} - \bar{u}_{t+h\Delta t} \|_1 \leq h \left( \sum_{k=1}^{K} \varepsilon_k + \varepsilon_c \right) + \varepsilon. \]

**Remark 4.5.** By the linearity of the operator \( e^{At} \) we observe that the \( \hat{u}_{t+h\Delta t} \) is an FSP approximation of \( p_{t+h\Delta t} \). That is, for \( x \in \Omega \)

\[ 0 \leq \hat{u}_{t+h\Delta t}(x) \leq p_{t+h\Delta t}(x). \]

We show the support of PFSP approximation has optimal order.

**Theorem 4.6.** Let \( \bar{u}_{t+h\Delta t} \) be the PFSP approximation of the CME problem (4.2.1.1), with the input parameters as given in Table 4.1. Let \( p_{t+h\Delta t} \) be the real solution to \( \bar{u}_{t+h\Delta t} \) with

\[ \| p_{t+h\Delta t} - \bar{u}_{t+h\Delta t} \|_1 \tau, \quad (4.18) \]

for \( \tau > 0 \). Let the last compression step have error \( \tau \delta \), where \( \delta \in (0, 1) \), and

\[ \| p_{t+h\Delta t} - \hat{u}_{t+h\Delta t} \|_1 = \tau(1 - \delta). \]

If \( p_{t+h\Delta t} \in \ell_{r,\infty} \) for an \( r \in (0, 1) \), then

\[ | \text{supp} \bar{u}_{t+h\Delta t} | \leq C \| p_{t+h\Delta t} \|_{r,\infty}^{1/s} (\tau \delta)^{-1/s}, \quad (4.19) \]

for \( C > 0 \) and \( \frac{1}{r} = s + 1 \).

**Proof.** The approximation \( \hat{u}_{t+h\Delta t} \) is an FSP approximation of \( p_{t+h\Delta t} \), hence Theorem 2.12 states

\[ 0 \leq \hat{u}_{t+h\Delta t}(x) \leq p_{t+h\Delta t}(x), \quad \forall x \in \Omega. \]

Applying Lemma 3.3 we get

\[ \| \hat{u}_{t+h\Delta t} \|_{r,\infty} \leq \| p_{t+h\Delta t} \|_{r,\infty}. \quad (4.20) \]

Since \( \bar{u}_{t+h\Delta t} \) is the best \( N \)-term approximation of \( \hat{u}_{t+h\Delta t} \), Lemma 3.7 gives us

\[ | \text{supp} \bar{u}_{t+h\Delta t} | \leq C_1 \| \hat{u}_{t+h\Delta t} \|_{r,\infty}^{1/s} (\tau \delta)^{-1/s}, \quad (4.21) \]

with \( C_1 > 0 \). Applying (4.20) gives us the optimal order result we need as follows,

\[ | \text{supp} \bar{u}_{t+h\Delta t} | \leq C \| p_{t+h\Delta t} \|_{r,\infty}^{1/s} \tau^{-1/s}, \]

for some \( C > 0 \). \( \Box \)

\footnote{This result uses Conjecture A in the proof.}
CHAPTER 4. THE PARALLEL FINITE STATE PROJECTION METHOD (PFSP)

4.5 Numerical Experiments

4.5.1 Realisations Vs Minimal Support

In the following experiment, we look at the number of realisations required to generate an empirical distribution with a prescribed accuracy. We then compare the number of realisations to the support size of an OFSP approximation of the same prescribed error.

We use the Poisson distribution as it is the simplest example with replicable results. The Poisson distribution is given for \( x \in \mathbb{N} \) as,

\[
P(x, t) = \frac{(\lambda t)^x e^{-\lambda t}}{x!}.
\]

Let \( \lambda = 10 \) and let \( t = 30 \). Table 4.2 gives the number of realisations and their error.

<table>
<thead>
<tr>
<th>Exact</th>
<th>Minimal Support</th>
<th>Error</th>
<th>Compression %</th>
</tr>
</thead>
<tbody>
<tr>
<td>133</td>
<td>0</td>
<td>0.001</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OFSP Support Size</th>
<th>Error</th>
<th>Compression %</th>
</tr>
</thead>
<tbody>
<tr>
<td>365</td>
<td>0.001</td>
<td>36%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSA Realisations</th>
<th>Error</th>
<th>Compression %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.234</td>
<td>11.3%</td>
</tr>
<tr>
<td>10000</td>
<td>0.0761</td>
<td>1.3%</td>
</tr>
<tr>
<td>50000</td>
<td>0.0321</td>
<td>0.22%</td>
</tr>
<tr>
<td>( 10^5 )</td>
<td>0.025</td>
<td>0.13%</td>
</tr>
<tr>
<td>( 2 \times 10^5 )</td>
<td>0.014</td>
<td>0.05%</td>
</tr>
<tr>
<td>( 10^6 )</td>
<td>0.0007</td>
<td>0.013%</td>
</tr>
</tbody>
</table>

We see in the results that stochastic simulation methods will need many more realisations to compute an approximation of the same error as the OFSP. As the distribution which we try to approximate spreads and flattens, the required number of realisations increase. Likewise, if the distribution is concentrated, then the number of realisations decrease. However, we show in Theorem 4.1 that even for a concentrated distribution, the minimal support is smaller than the number of realisations of a stochastic simulation.
4.5. NUMERICAL EXPERIMENTS

4.5.2 PFSP Example

We compute a simple 20 dimensional mono-molecular RDME problem to demonstrate the implementation of a PFSP method.

Consider a system with 20 compartments in serial. The system has one particle type, denoted as $A$. Particle $A$ undergoes decay, birth and diffusion between compartments.

We simplify this system to one with 20 different particles, denoted by $A_i$ for $i = 1, \ldots, 20$. We denote the population as $A_i$ in the $i$th compartment by $A_i$. The propensities and stoichiometry are given in Table 4.3. We compute the model with an initial condition, $x_0 = (\delta(n))_{n=1}^{20}$, until $t_{final} = 1$. We choose a uniform time step of $\Delta t = 0.01$. The PFSP method was implemented using CMEPy and was run on a Quad-Core AMD Opteron(tm) Processor 8360 SE with 4GB of RAM.

<table>
<thead>
<tr>
<th>Reaction</th>
<th>Propensity</th>
<th>Stoichiometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>$* \rightarrow A^i, i = 1, \ldots, 20$</td>
<td>$\alpha_1((A^i)_{i=1}^{20}) = 0.01$</td>
<td>$(\delta(n))_{n=1}^{20}$</td>
</tr>
<tr>
<td>$A^i \rightarrow *, i = 1, \ldots, 20$</td>
<td>$\alpha_2((A^i)_{i=1}^{20}) = 0.05[A^i]$</td>
<td>$(-\delta_i(n))_{n=1}^{20}$</td>
</tr>
<tr>
<td>$A^i \rightarrow A^{i+1}, i = 1, \ldots, 19$</td>
<td>$\alpha_3((A^i)_{i=1}^{20}) = 0.25[A^i]$</td>
<td>$(-\delta_i(n) + \delta_{i+1}(n))_{n=1}^{20}$</td>
</tr>
<tr>
<td>$A^{i+1} \rightarrow A^i, i = 1, \ldots, 19$</td>
<td>$\alpha_4((A^i)_{i=1}^{20}) = 0.25[A^{i+1}]$</td>
<td>$(\delta_i(n) - \delta_{i+1}(n))_{n=1}^{20}$</td>
</tr>
</tbody>
</table>

Table 4.3: 20D mono-molecular system propensity and stoichiometry

<table>
<thead>
<tr>
<th>$t_{final} = 1$</th>
<th>Number of states/realisation</th>
<th>Error</th>
<th>Comp* time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFSP ($h = 2$)</td>
<td>$\approx 300000$</td>
<td>0.1</td>
<td>14822</td>
</tr>
<tr>
<td>PFSP ($Cores = 3, h = 5$)</td>
<td>336709</td>
<td>0.1</td>
<td>1949</td>
</tr>
<tr>
<td>PFSP ($Cores = 4, h = 5$)</td>
<td>249860</td>
<td>0.1</td>
<td>934</td>
</tr>
<tr>
<td>PFSP ($Cores = 8, h = 5$)</td>
<td>476441</td>
<td>0.1</td>
<td>887</td>
</tr>
<tr>
<td>SSA ($Cores = 12$)</td>
<td>$10^6$</td>
<td>$\approx 0.8$</td>
<td>1348</td>
</tr>
</tbody>
</table>

Table 4.4: Computation times of the SSA, PFSP and OFSP method.

In the implementation of OFSP, the approximation is compressed every two steps. For the PFSP method, a compression was performed every three steps on each core. The PFSP method collates all state spaces onto a single core every five steps, aggregating similar states and redistributing the problem over multiple cores. The partitioning and distribution of the domain onto multiple cores is non-linear.
Chapter 4. The Parallel Finite State Projection Method (PFSP)

The PFSP method used to compute the example is implemented using CMEPy [1]. An OFSP method is computed on each core. The domain is partitioned in such a way that each partition has the same number of states and a similar amount of probability. The implementation of the particular example above is successful in reducing the computation time. However, this is not the case with all problems, specifically for systems with large number of reactions ($\gg 60$). This is a restriction due to the implementation, rather than the method. This is discussed below in section 4.6.

4.6 Discussion

The focus of this chapter is on the presentation of a CME solver to solve RDME models. Conventionally, computing approximations to RDME models using an FSP solver is not preferred, the major concern being the large hyper-rectangular domain that might need to be constructed for an approximation. We have shown however, that the number of realisations required to compute approximations as accurate as the best finite state projection (minimal support, Theorem 4.1) approximation is larger. We have also shown empirically that it is significantly large. Combined with an efficient algorithm to find the domain, the FSP method can be several order of magnitudes faster than the stochastic simulation methods. We propose such a method in section 5.

A major computational advantage of stochastic simulation methods are that they can be easily parallelised with a near linear speed-up expected. We thus presented the Parallel Finite State Projection method to parallelise the OFSP algorithm. In the numerical experiment conducted above, the PFSP method was nearly ten times as fast. However, to replicate these results across all problems, some key implementation details must be considered. The current implementation is simply an extension of the OFSP code to multiple cores. Addressing the following points would result in a faster solver.

Firstly, most of the computation time in any FSP implementation is spent on finding/constructing the domain. This implies that there is a significant amount of memory access, specifically, small amounts of memory being accessed at high frequencies. The current domain finding/constructing methods waste clock-cycles due to the high frequency of RAM access, which is a result of the small size of the much faster L2 cache. To overcome this, the search problem has to be subdivided into blocks, with each block of data stored on the L2 cache on completion of
the CPU operation. With the present architecture used in most HPC clusters, each machine has multiple processors, with each processor having multiple cores. Each processor has an L2 cache shared among its cores. If a thread running on a single core requires access to memory outside the L2 cache, it is forced to queue up the request with those from other threads running on other cores. This forces the thread to waste a significant number of clock-cycles waiting for data from RAM. Thus, memory management is a key point to be addressed in a future implementation.

Secondly, the search for the next states to be added to the domain can be done in parallel. Distributing the task of finding the next states in the state space to the GPU or to idle cores would result in a significant speed-up. Most HPC infrastructures to date have a GPU chipset on each blade which can be utilised to speed up computation.

Lastly, the current implementation of the PFSP uses a master-slave hierarchy, where all information is sent up to the master core. The master core constructs sub-problems and the slave cores are then provided with these problems to process them. While this is a simple implementation, there is a significant bottleneck since some cores finish their computation and sit idly, waiting for other cores to finish. The construction of a different hierarchy, where all cores are productive while waiting for slower cores would result in a speed-up.

In the next chapter we present the Gated One Reaction Domain Expander (GORDE), the first domain finding algorithm which is proven to output the domain with the prescribed error. To prove the functionality of the GORDE algorithm, we revisit the CME with respect to reaction counts.
Chapter 5

Gated One Reaction Domain Expansion

Gated One Reaction Domain Expander (GORDE) is an algorithm for adaptively growing the domain in regions which increase the accuracy of the FSP type approximations. The method is titled the “Gated One Reaction Domain Expander”, as the method only allows a single reaction to occur in a given time period. By only allowing one reaction, the probability of one reaction occurring forms an upper bound on the original probability. The upper bounds are then used to determine which states are to be added or removed. In section 3 we extensively discuss the importance of computing over the smallest possible domain rather than constructing a hyper-rectangular domain. By computing near the optimal support size using the OFSP, a speed up of a few orders of magnitude can be achieved (§3.3). However, the OFSP method guarantees order optimality post computation, that is, only after the compression step is the support of optimal order. To achieve a faster computation time, the next step is to find the minimal support a priori. When growing/adding new states to compute over, only the minimal number of states should be added.

For finding the minimal domain in the state space, a metric needs to be specified over the state space to introduce a bias that one state is more important than another. The problem is choosing/constructing a metric such that it guarantees that the minimal domain or near minimal domain is found. The N-step method, introduced by Munsky and Khammash in their paper on the FSP method is the first and most popular method for domain selection [49]. The N-step method is based on the logic that the next state to select should only be a state which is reachable from the current state via a linear combination of reactions. Con-
structively, the N-step method introduces all the states which are one reaction away from the existing domain, followed by those which are two reactions away and so on. This is repeated incrementally until the required error on the FSP approximation is satisfied. In the N-step method the metric value is one if the state is reachable by the specified number of reactions and zero otherwise. Munsky and Khammash show that for an error there is a finite number of reactions, which when computed, give us an approximation with the right error. The N-step method metric gives the same importance for each state which is reachable via a sequence of reactions. Since states can be reachable, but significantly improbable, the N-step method adds redundant states. The N-step method is very successful in FSP implementations, especially for smaller dimensions, the simpler metric results in faster computation time as the order of the number of states we are adding is small. Hence the computation of redundant states has a negligible impact on computation time. The N-step method is further discussed in section 2.3.2.

Another method for domain selection is the Sliding Window method (SW), proposed by Wolf et. al [88]. The SW method computes a number of stochastic simulations from random points in the existing domain, to estimate which states are in the future state space. The SW method takes the coordinate-wise maximum and minimum of all the realisations and then outputs the new domain to be a hyper-rectangle with corners as realised maximum and minimum. The convergence of the SW method is a function of the number of realisations computed to estimate the corners of the hyper-rectangle. Wolf et. al claim that for general problems the accuracy of the domain choice does not increase after taking 100 realisations [88]. Taking realisations to estimate the location of the domain is advantageous when we have multiple reactions with different time scales. If the system is firing a particular reaction faster than other reactions, the state space would be skewed in the direction of the states which are a consequence of that fast reaction. This directional growing strategy, of the SW, proves effective for stiff CME problems. The stiffness arises when the system has fast reactions and slow reactions, for these systems the SW method will grow the domain more in the direction of the fast reactions firing. For the Göutsias problem, the SW method attains a speed up of two orders [88] of magnitude to the method of operator splitting and N-stepping. However a problem of the SW method is that it can remove key features of a probability distribution, especially when the CME solution is a multiple modal distribution. Consider the example where the solution to the CME is mono-modal in one time step and in the next time step it
has two modes. If all the realisations by the SW method are in the domain of the first mode, then the SW method removes the second mode. This problem is specific to the class of problems with multiple modes. The SW method cannot be extended to large CME or RDME problems since constructing hyper-rectangles is an unfeasible exercise, as discussed in section 3. The SW method is discussed in further detail in section 2.3.3.

In summary, the $N$-step method uses the correct logic of searching along possible reactions that occur, however its uniform priority to all the states results in the addition of states which are redundant, making large dimensional problems difficult to compute. The SW method uses the stochastic simulations to give a better indication of which states are important. However, it’s method of constructing hyper-rectangles is unfeasible for high-dimensional CME problems. The problem in domain selection is that of accuracy; both methods will converge, however their accuracy is only found by trial and error. For both methods, the accuracy of the solution is found after computation, if the method exists with less accuracy than expected, the algorithms are repeatedly applied until the error is above the desired accuracy. This is an expensive exercise. The trial and error method forces the solvers to take small time steps, in practice $\Delta t$ is inversely proportional to the largest propensity. Choosing such a small step helps reduce the number of trials needed to compute the right answer. However, it increases the number of states added and the number of computations. The problem is to find an algorithm which will converge and without doing a trial step, the accuracy of the domain size selection is known a priori to computing the CME problem. In this chapter, the Gated One Reaction Domain Expander (GORDE) is introduced for the first time, the method guarantees convergence and accuracy a priori to computing the CME problem.

In the following sections the solution of the CME with respect to reaction counts by Sunkara [80] is visited. The GORDE algorithm is then presented and its convergences and accuracy are proved by expanding the reaction counts framework.

### 5.1 CME with Respect to Reaction Counts

This section is a review of the framework and results given by Sunkara in 2009 [80]. The CME describes biological systems where different species are undergoing reactions. The main focus is on computing the probability of the different
configurations of population counts that the system undergoes. Kurtz, in 1972, suggested to count the number of reactions fired rather than counting the change in population, and that doing this gives an equivalent formulation \[53\]. The species population only change due to the firing of reactions. Hence, by knowing the original population and counting the reactions which have fired, the population changes can be found \[39, 86\]. The structure of the reaction counts is intrinsically simpler than that of the species counts, that is, reactions either occur or they do not, they cannot “de-fire”. Thus, the state space for the reaction counts is connected in a forward direction. If there are \(N_r\) reactions, then each reaction state can only be reached via \(N_r\) adjacent points. Sunkara in 2009 showed that the CME with respect to reaction counts had a simple structure and that this structure gave rise to explicit solutions and eigenvalues. The formulation of the CME with respect to reaction counts and it’s key results are presented below.

Given a system with \(N_s\) species undergoing \(N_r\) reactions, let \(\Omega \subset \mathbb{N}_0^{N_s}\) be the set of all the states reachable by the system starting at an initial state \(x_0 \in \mathbb{N}_0^{N_s}\). We denote the species configuration by \(x \in \mathbb{N}_0^{N_s}\) and the reaction configuration by \(r \in \mathbb{N}_0^{N_r}\).

We define \(V = [v_1, \ldots, v_{N_r}]^T \in M_{N_r \times N_s}\) to be the stoichiometric matrix, where \(v_i\) are the stoichiometric vectors defined in §2.1.1. For a fixed \(x_0 \in \Omega\), using the stoichiometric matrix define an affine map \(\Gamma_{x_0} : \mathbb{N}_0^{N_r} \to \mathbb{N}_0^{N_s}\), by

\[
\Gamma_{x_0}(r) := x_0 + V^T r, \quad r \in \mathbb{N}_0^{N_r}. \tag{5.1}
\]

The map \(\Gamma_{x_0}\) describes the point at which the population of species would be, if the population started at \(x_0\) and reactions \(r\), occurred. If a system starts with population \(x_0\), let

\[
\Lambda := \{r \in \mathbb{N}_0^{N_r} : \Gamma_{x_0}(r) \in \Omega\}.
\]

Hence \(\Lambda\) is the set of different reaction configurations that the systems can undergo. There is an affine relationship between the reactions firing and the population configuration that the system is in. Hence the aim is to define a probability distribution over the configuration of reactions firing and then to use the affine map to give the probability distribution over the population configurations.

Given a system starting with population configuration of \(x_0 \in \Omega\). Let \(\alpha_i\) for \(i = 1, \ldots, N_r\) be the propensities function of the \(i\)th reaction and have the property
5.1. CME WITH RESPECT TO REACTION COUNTS

\[ \alpha_i(x) := \begin{cases} \geq 0, & \text{if } x \in \Omega, \\ 0, & \text{otherwise}. \end{cases} \]

Let \( \bar{P}(r, t) \) denote the probability of a configuration of reactions, \( r \), firing by time \( t > 0 \). For \( r \in N_0^{N_r} \) and \( t > 0 \), define \( \bar{P}(r, t) \) as the solution of the following equations:

\[
\frac{\partial \bar{P}(0; t)}{\partial t} = - \sum_{j=1}^{N_r} \alpha_j(\Gamma_{x_0}(0)) \bar{P}(0; t),
\]

(5.2)

and

\[
\frac{\partial \bar{P}(r; t)}{\partial t} = \sum_{j=1}^{N_r} \alpha_j(\Gamma_{x_0}(r - I_j)) \bar{P}(r - I_j; t) - \alpha_j(\Gamma_{x_0}(r)) \bar{P}(r; t),
\]

(5.3)

where \( I_j \) is unit vector with 1 in the \( j \)th entry. For each \( x \in \Omega \), define the set

\[
\Gamma^{-1}_x := \{ r \in N_0^{N_r} : \Gamma_{x_0}(r) = x \}.
\]

(5.4)

For a system starting at \( x_0 \in \Omega \), using the push forward measure, we can form the link that, for all \( x \in \Omega \),

\[
P(x; t) = \sum_{r \in \Gamma^{-1}_x} \bar{P}(r; t).
\]

(5.5)

The description above states that the probability of the system being in a population configuration at a time \( t \) is the sum over the probabilities of reaction configurations which lead to that particular population configuration.

We know from Theorem 2.8 that the right-hand side terms of (5.5) are all positive. If the probability distribution in the reaction counts has norm less than or equal to one, we know (5.5) holds via the push forward measure. Since there are multiple reactions which can all give the same population configuration, the total sum of probability of these reactions give us the probability of being in that population configuration. Hence, the reaction count probabilities converges to the species counts probability from below. It is difficult to exactly indicate the rate at which they converge. If we were to map the reaction states corresponding to a particular population configuration onto the natural numbers, using the \( \ell_1 \) norm, we would see a bell shaped curve with the peek at the most probable reaction configuration. Over time, this peek would travel down the natural numbers. Later in this Chapter we introduce the GORDE algorithm which estimates how
many of reaction counts states have to be considered to get a particular $\ell_1$ error in the species count.

For a system, computing the probability of a reaction occurring, (5.3), and then summing the probabilities of reactions to give the probability of the population configuration, (5.5), is referred to as solving the CME with respect to reaction counts.

Similar to (2.15), the solution to (5.3) is found by solving the following linear differential equation;

$$\frac{dP(t)}{dt} = A_rP(t),$$

(5.6)

where $P(t)$ is the vector in $[0, 1]^{||A||}$ with coordinates $\bar{P}(r,t)$, $r \in \Lambda$. The initial condition is that $\bar{P}(\cdot,t) = \delta_0(\cdot)C$, with $C \in (0, 1]$. In this formulation the $A_r$ can be rearranged to be lower triangular, which is not always the case for $A$, given in (2.15) of the species counts. Having $A_r$ be lower triangular allows the derivation of some properties of the point spectrum of $e^{A_r t}$ and also explicit solutions for (5.2) and (5.3). We derive these results below.

**Remark 5.1.** If $\Gamma_{x_0}$ is bijective, then solving the CME with respect to reaction counts is the same as solving the original CME. We can expect $\Gamma_{x_0}$ to be bijective when $V^T$ is invertible. If $V^T$ is rectangular we can see from linear algebra that $|\Gamma_x^{-1}|$ will be infinite for all $x \in \Omega$.

**Theorem 5.2.** Given a system with $N_s$ species and $N_r$ reactions with stoichiometric matrix $V$, if $\Lambda \neq \emptyset$, then

1. there exists a permutation $Pr$ such that $Pr^T A_r Pr$ is lower triangular,

2. the spectrum of $A_r$ is,

$$\text{spec}(A_r) = \{-\sum_{j=1}^{N_r} a_j(x_0 + V^T r) : r \in \Lambda \},$$

where $x_0$ is the starting population configuration of the system.

**Proof.** For a fixed $r \in \Lambda$, from (5.3) we can deduce that $\bar{P}(r; t)$ is only dependent on $\bar{P}(r_1; t)$ where $r_1 \in \Lambda$ with $\|r_1\|_1 < \|r\|_1$. Thus, the elements of $P$ can be rearranged such that $A_r$ is lower triangular. Since $A_r$ is lower triangular, its eigenvalues are the diagonal elements. The diagonal elements from (5.3) are the following set, \{-\sum_{j=1}^{N} a_j(x_0 + V^T r) : r \in \Lambda \}. \blacksquare
Corollary 5.3. If there exist distinct \( r_1, r_2 \in \Lambda \subset \mathbb{N}_0^{N_r} \) such that \( r_1, r_2 \in \Gamma_x^{-1} \) (c.f. (5.4)) for an \( x \in \Omega \), then \( A_r \) has non unique eigenvalues.

Proof. If there exists an \( x \in \Omega \) such that \( |\Gamma_x| \geq 2 \), then for any \( r_1, r_2 \in \Gamma_x \subset \Lambda \),

\[
\sum_{j=1}^{N} a_j(x_0 + V^T r_1) = \sum_{j=1}^{N} a_j(x_0 + V^T r_2).
\]

Hence the two eigenvalues corresponding with \( r_1 \) and \( r_2 \) given in Theorem 5.2 are the same.

If \( \text{rank} \ V < N_r \), then by Corollary 5.3 \( A_r \) has non-distinct eigenvalues. If \( V \) is invertible, then the eigenvalues of \( e^{A_r} \) are the same as \( e^{A_r} \). If they are not, we cannot say much about the specific eigenvalues of the species counts. However, this is already a known result [24].

Proposition 5.4. Given a system with \( N_s \) species and \( N_r \) reactions with stoichiometric matrix \( V \) and starting state \( x_0 \in \Omega \). Let \( \Gamma_{x_0} \) be defined as in (5.1). If \( \Lambda \neq \emptyset \) and \( \bar{P}(\cdot, 0) = C\delta_0(\cdot) \), where \( C \in (0, 1] \), then the solution for (5.3) is given by:

\[
\bar{P}(0; t) = e^{-\left(\sum_{j=1}^{N_r} a_j(x_0)\right)t} C, \ t > 0,
\]

and

\[
\bar{P}(r; t) = \sum_{j=1}^{N_r} \left( a_j(\Gamma_{x_0}(r - I_j)) \int_0^t e^{-\left(\sum_{j=1}^{N_r} a_j(\Gamma_{x_0}(r))\right)(t-s)} \bar{P}(r - I_j; s) ds \right),
\]

for \( r \in \Lambda \setminus \{0\} \) and \( t > 0 \).

Proof. Case 1: Let \( r = 0 \), then (5.3) reduces to;

\[
\frac{\partial \bar{P}(0; t)}{\partial t} = -\sum_{j=1}^{N_r} a_j(x_0) \bar{P}(0; t).
\]

This solves analytically to (5.7).

Case 2: Fix \( r \in \Lambda \setminus \{0\} \). Let \( g(t) := \sum_{j=1}^{N_r} a_j(\Gamma(r - I_j)) \bar{P}(r - I_j; t) \) and \( \beta = \sum_{j=1}^{N_r} a_j(\Gamma(r)) \). Then (5.3) is of the form,

\[
\frac{\partial \bar{P}(r; t)}{\partial t} = g(t) - \beta \bar{P}(r; t).
\]
The Laplace transform of (5.9) with \( \mathcal{L}(\bar{P}) \) and \( \mathcal{L}(g) \) is

\[
s\bar{P} - \bar{P}(r,0) = G(s) - \beta \bar{P}.\]

Since \( \bar{P}(r,0) = 0 \) by the initial condition, \( \bar{P}(\cdot,0) = C\delta(\cdot) \), and

\[
\bar{P} = \frac{1}{s + \beta} G(s). \tag{5.10}
\]

The inverse Laplace transform of (5.10) is,

\[
\bar{P}(r;t) = \int_0^t e^{-\beta(t-s)} g(s) ds.
\]

The re-substitution of \( g(t) \) and \( \beta \) gives (5.8).

**Example 5.5. Growth and Decay Process.** Let \( S \) be a particle undergoing two reactions:

\[
S \xrightarrow{0.8 \times |S|} \emptyset, \quad \emptyset \xrightarrow{0.4} S.
\]

The propensity of the Decay Process is given by

\[
\alpha_1(x) := \begin{cases} 
0.8x, & x \in \Omega, \\
0, & \text{otherwise},
\end{cases}
\]

and propensity of the Growth Process is given by

\[
\alpha_2(x) := \begin{cases} 
0.4, & x \in \Omega, \\
0, & \text{otherwise},
\end{cases}
\]

The stoichiometric matrix is \( \mathbb{V}^T = [-1,1] \). Let \( x_0 = 23 \). The full state space of this process is \( \mathbb{N}_0 \). The reaction state space is given in Figure 5.1(a) and the equivalence class of reactions states which aggregate under the affine map, \( \Gamma \) are given in Figure 5.1(b).

The CME with respect to reaction counts for the states \((0,0), (1,0), (0,1)\) and \((1,1)\) are

\[
\begin{bmatrix}
\partial_t \bar{P}((0,0); t) \\
\partial_t \bar{P}((0,1); t) \\
\partial_t \bar{P}((1,0); t) \\
\partial_t \bar{P}((1,1); t)
\end{bmatrix} =
\begin{bmatrix}
-18.8 & 0 & 0 & 0 \\
18.4 & -18 & 0 & 0 \\
0.4 & 0 & -19.6 & 0 \\
0 & 0.4 & 19.4 & -18.8
\end{bmatrix}
\begin{bmatrix}
\bar{P}((0,0); t) \\
\bar{P}((0,1); t) \\
\bar{P}((1,0); t) \\
\bar{P}((1,1); t)
\end{bmatrix}
\]

Proposition 5.4 gives the following solutions:
5.1. CME WITH RESPECT TO REACTION COUNTS

\[ P((0, 0); t) = e^{-18.8t}. \]

\[ \hat{P}((0, 1); t) = \alpha_1(\Gamma(0, 0)) \int_0^t e^{-\left(\alpha(\Gamma((0,1)))+\alpha(\Gamma(0,1))\right)(t-s)} \hat{P}((0, 0); s) ds, \]

\[ = 23e^{-18t} - 23e^{-18.8t}. \]

\[ \hat{P}((1, 0); t) = \alpha_2(\Gamma(0, 0)) \int_0^t e^{-\left(\alpha(\Gamma((0,1)))+\alpha(\Gamma(0,1))\right)(t-s)} \hat{P}((0, 0); s) ds, \]

\[ = -0.5e^{-19.6t} + 0.5e^{-18.8t}. \]

\[ \hat{P}((1, 1); t) = \sum_{i=1}^{2} \alpha_i(\Gamma((1,1)) - I_i) \int_0^t e^{-\left(\alpha(\Gamma((1,1)))+\alpha(\Gamma(1,1))\right)(t-s)} \hat{P}((1, 1) - I_i; s) ds \]

\[ \approx 11.5e^{-18t} + 12.12e^{-19.6t} - 23.63e^{-18.8t}. \]

Mapping these reaction states back to species population by (5.5),

\[ P(23; t) > \hat{P}((0, 0); t) + \hat{P}((1, 1); t), \ P(22; t) > \hat{P}((1, 0); t), \ P(24; t) > \hat{P}((0, 1); t). \]

**Example 5.6 (Decay Process).** Let \( X(t) \) be a stochastic process describing the discrete decay of a collection of particles. At time zero \( X(0) = 3 \) and the propensity of the decay is given by

\[ \alpha(x) := x. \]

The state space for this process is simply \( \Omega = \{0, 1, 2, 3\} \) and the infinitesimal generator is given by,
CHAPTER 5.  GATED ONE REACTION DOMAIN EXPANSION

\[
A := \begin{bmatrix}
-3 & 0 & 0 & 0 \\
3 & -2 & 0 & 0 \\
0 & 2 & -1 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix},
\]

and the semi-group,

\[
e^{At} = \begin{bmatrix}
e^{-3t} & 0 & 0 & 0 \\
-3e^{-3t} + 3e^{-2t} & e^{-2t} & 0 & 0 \\
-6e^{-2t} + 3e^{-3t} + 3e^{-t} & -2e^{-2t} + 2e^{-t} & e^{-t} & 0 \\
-3e^{-t} + 3e^{-2t} - e^{-3t} + 1 & e^{-2t} + 1 - 2e^{-t} & 1 - e^{-t} & 1
\end{bmatrix}.
\]

Applying the initial condition we get the PDF,

\[
P(3; t) = e^{-3t} \\
P(2; t) = -3e^{-3t} + 3e^{-2t} \\
P(1; t) = -6e^{-2t} + 3e^{-3t} + 3e^{-t} \\
P(0; t) = -3e^{-t} + 3e^{-2t} - e^{-3t} + 1
\]

(5.11)

We now use Proposition 5.4 to compare the probability distribution.

\[
P(3; t) = \bar{P}(0; t) = e^{-3t}.
\]

\[
P(2; t) = \bar{P}(1; t) = 3 \int_0^t e^{-2(t-s)} P(3; s) ds,
\]

\[
= 3e^{-2t} \int_0^t e^{2s} e^{-3s} ds,
\]

\[
= 3e^{-2t} \left[ -e^{-s} \right]_0^t,
\]

\[
= 3(e^{-2t} - e^{-3t}).
\]
\[ P(1; t) = \bar{P}(2; t) = 2 \int_0^t e^{-(t-s)} P(2; s) ds, \]
\[ = 6 \int_0^t e^{-(t-s)} (e^{-2s} - e^{-3s}) \, ds, \]
\[ = 6e^{-t} \left( [-e^{-s}]_0^t + \left[ \frac{e^{-2s}}{2} \right]_0^t \right), \]
\[ = -6e^{-2t} + 3e^{-3t} + 3e^{-t}. \]

\[ P(0; t) = \bar{P}(3; t) = \int_0^t P(1; s) ds, \]
\[ = \int_0^t -6e^{-2s} + 3e^{-3s} + 3e^{-s} \, ds, \]
\[ = -6 \left[ \frac{e^{-2s}}{-2} \right]_0^t + 3 \left[ \frac{e^{-3s}}{-3} \right]_0^t + 3 \left[ \frac{e^{-s}}{-1} \right]_0^t, \]
\[ = -3e^{-t} + 3e^{-2t} - e^{-3t} + 1. \]

Thus, the solution from solving the matrix exponential matches the solution found by computing Proposition 5.4.

The CME with respect to reaction counts can be solved with less computational complexity than the original CME. However, if the stoichiometric matrix is non-invertible, then the solution of the reaction counts will converge to the solution of the species counts from below. In most cases the stoichiometric matrix is non-invertible, hence computing the reaction counts cannot guarantee quick computation. The convergence of the reaction count solutions to the species counts solutions motivated their use for domain estimation. The next section presents the GORDE algorithm, which utilises the reaction count framework to accurately estimate the domain for a prescribed error at each time step of the CME problem.

## 5.2 GORDE Algorithm

In Section 3 it is discussed that for large dimensional problems constructing a hyper-rectangle and and solving the CME inside it is unfeasible. Hence, to successfully solve the CME problem, the solver is expected to construct the domain as needed. In these solvers, the domain finding algorithms are important. In
practice, successfully finding the domain is the most intense aspect of computing
the solutions to the CME. The Sliding Window (SW) [88] and the N-step [49]
methods are the state of the art methods available to determine the domain.
While both methods converge, they do not have a stopping criteria which guar-
antees that the required domain is found. That is, in implementation, a trial
evaluation of the CME problem over the proposed domain is conducted. If the
error of the approximation is within prescribed tolerance, then the the domain
has been found, otherwise the algorithms are computed again. To avoid too many
trial evaluations, the solvers are set to take time steps inversely proportional to
the largest propensity evaluation in the domain. Selecting that time step ensures
that only a small number of reactions have fired, since the inverse proportional
of the largest propensity is the time taken for a single reaction to occur. The
GORDE algorithm has a stopping criterion which guarantees that after stop-
ping, the domain proposed by the method produces an approximation which is
of the right error. The GORDE algorithm is presented in Algorithm [10].

<table>
<thead>
<tr>
<th>Table 5.1: GORDE input parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_t$ Starting vector.</td>
</tr>
<tr>
<td>$\Omega_t$ The support of $w_t$.</td>
</tr>
<tr>
<td>$\Delta t$ The time step to find the support for.</td>
</tr>
<tr>
<td>$\varepsilon &gt; 0$ The amount of probability willing to lose</td>
</tr>
<tr>
<td>$N_r$ The total number of reactions</td>
</tr>
<tr>
<td>$\alpha_i$ $i = 1, \ldots N_r$, the propensity of $i$th reaction.</td>
</tr>
<tr>
<td>$\alpha_0$ $\alpha_0 := \sum_{i=1}^{N_r} \alpha_i$</td>
</tr>
<tr>
<td>$\nu_i$ $i = 1, \ldots N_r$, the stoichiometric vector of the $i$th reaction.</td>
</tr>
<tr>
<td>$R_i$ $R_i(x) := x + \nu_i$ where $x \in \Omega$.</td>
</tr>
</tbody>
</table>
Algorithm 10: GORDE Algorithm

**input**: \( w_t, \Omega_t, \Delta t, \varepsilon, N_r, \alpha_0, \ldots, \alpha_{N_r} \)

**output**: \( \Omega_{t+\Delta t} \)

1. \( \nabla_0 \leftarrow \Omega_t \)
2. \( u_0 \leftarrow w_t \)
3. \( \tau_0 \leftarrow \varepsilon \)
4. \( m \leftarrow 1 \)
5. while \( |\nabla_{m-1}| > 0 \) do
6.  
7.     for \( i \in 1, \ldots, N_r \) do
8.         \( \nabla_{m,i} \leftarrow R_i(\nabla_{m-1}) \)
9.         for \( x \in \nabla_{m,i} \) do
10.             \( u_{m,i}(x) \leftarrow \frac{\alpha_i(x-v_i)}{\alpha_i(x-v_i)}(1 - e^{-\alpha_0(x-v_i)\Delta t})u_{m-1}(x-v_i) \)
11.         end
12.     end
13.     \( \nabla_m \leftarrow \bigcup_{i=1}^{N_r} \nabla_{m,i} \)
14.     for \( x \in \nabla_m \) do
15.         \( u_m(x) \leftarrow \sum_{i=1}^{N_r} u_{m,i}(x) \)
16.     end
17.     \( \nabla_m, u_m, \tau_m \leftarrow \text{reduce}(u_m, \nabla_m, \tau_{m-1}) \)
18.     \( m \leftarrow m + 1 \)
19. end
20. return \( \Omega_{t+\Delta t} \leftarrow \bigcup_{j=0}^{m-1} \nabla_j \)
21. end

The \textit{reduce} function mentioned in the algorithm above is a best \( N \)-term approximation with the residual and the support included in the function return values.

The GORDE algorithm can be verbosely described as following: Given a probability distribution, \( p \), with support, \( \Omega \), we wish to find the support of the evolution of \( p_t \) by \( \Delta t > 0 \) with an FSP error of \( \varepsilon \). The GORDE algorithm takes the following steps to find the new support: Firstly, the method considers all the states that can be reached via a single reaction from the current support, \( \tilde{\nabla}_1 = \bigcup_{i=1}^{N_r} R_i(\Omega) \), then each \( x \in \nabla_1 \) is given a positive value \( u_1(x) \in [0, 1) \). We then remove all the \( x \in \nabla_1 \subset \tilde{\nabla}_1 \), where \( \tilde{\nabla}_1 \) is the biggest set which satisfies, \( \sum_{x \in \nabla_1} u_1(x) \leq \varepsilon \). If \( \tilde{\nabla}_1 = \tilde{\nabla}_1 \), then \( \Omega \cup \tilde{\nabla}_1 \) is the future support, the evolution of \( p \) on this support for a step, \( \Delta t \), will give an FSP error of less than \( \varepsilon \). However, if
92  

\[ \nabla_1 \neq \tilde{\nabla}_1, \text{ then we take another iteration now with the support at } \nabla_1 := \nabla_1 \setminus \tilde{\nabla}_1 \text{ and the distribution as } u_1|_{\nabla_1} \text{ with the new threshold } \tau_1 := \varepsilon - \sum_{x \in \nabla_1} u_1(x). \text{ Hence we compute a new support } \nabla_2 \text{ and compute new values } u_2. \text{ If } \sum_{x \in \tilde{\nabla}_2} u_2(x) \leq \tau_1, \text{ we compute the future support for error } \varepsilon \text{ is } \Omega + \nabla_1 + \nabla_2. \text{ If the criterion is not met then another iteration is taken. Let } n \in \mathbb{N} \text{ be the iteration at which } \sum_{x \in \tilde{\nabla}_n} u_n(x) \leq \tau_{n-1}, \text{ then the new support is given by,} \]

\[ \Omega \cup \nabla_1 \cup \cdots \cup \nabla_{n-1} \cup \tilde{\nabla}_n. \]

The \( u_m \) in the GORDE algorithm is derived from an approximation which we call *gating*. Gating is the solution of the CME of a state which can be reached via a reaction. However no reaction can remove it from that state. Using the linearity of the CME operator, we find that a gated state is always an upper bound with respect to probability of the state not being gated. In Step 10 of Algorithm 10, \( u_m(x) \) is the gating solution, and is given by,

\[ u_m(x) = \sum_{i=1}^{N} \frac{\alpha_i(x - v_i)}{\alpha_0(x - v_i)} (1 - e^{-\alpha_0(x-v_i)\Delta t})u_{m-1}(x - v_i), \]

for \( m \in \mathbb{N} \) and \( x \in \nabla_m \). In section 5.2.1 we give the physical meaning of this approximation and the accuracy and convergences of this selection.

**Example 5.7.** The following example illustrates the GORDE algorithm for a simple problem.

Let a system have two particles, particle \( A \) and particle \( B \), which can undergo two reactions \( R_1 \) and \( R_2 \) in the following way: \( A \overset{\alpha_1}{\rightarrow} B \) and \( B \overset{\alpha_2}{\rightarrow} A \). In this system \( \alpha_1 \) and \( \alpha_2 \) are the respective propensity functions of the reactions. For simplicity, a constant value of \( \alpha_1 := 0.5 \) and \( \alpha_2 := 0.7 \) are chosen. The stoichiometric vectors for the reactions are \( v_1 = (-1, 1) \) and \( v_2 = (1, -1) \) respectively. The state shifts for the reactions are given by \( R_1((A, B)) := (A - 1, B + 1) \) and \( R_2((A, B)) := (A + 1, B - 1) \).

The system is restricted to having only two particles at any given time. Therefore, the complete state space has three states

\[ \Omega = \{(2, 0), (1, 1), (0, 2)\}. \]

The initial condition the states \( (2, 0) \) and \( (1, 1) \) starting with equal probability of 0.5.
5.2. GORDE ALGORITHM

Then the CME problem (5.1.1) for this example is

\[
\begin{pmatrix}
\partial_t p_t((2,0)) \\
\partial_t p_t((1,1)) \\
\partial_t p_t((0,2))
\end{pmatrix} = \begin{pmatrix}
-\alpha_1 & \alpha_2 & 0 \\
\alpha_1 & -\alpha_1 - \alpha_2 & \alpha_2 \\
0 & \alpha_1 & -\alpha_2
\end{pmatrix} \begin{pmatrix}
p_t((2,0)) \\
p_t((1,1)) \\
p_t((0,2))
\end{pmatrix}
\] (5.12)

In matrix form we write

\[
\frac{dp_t}{dt} = Ap_t, \text{ with i.c. } p_0 = (0.5, 0.5, 0).
\]

The problem is to estimate the support at time \(\Delta t = 0.8\) which generates an FSP error of \(\varepsilon = 0.1\). At \(t = 0.8\) the probability distribution is \(p_{0.8} = (0.522, 0.354, 0.123)\). For an FSP approximation with the states in \(\Omega_0\), the approximation would have an error of more than 0.1, since \(p_{0.8}((0,2)) = 0.123\). The following are the steps of the values evaluated inside the GORDE method to find the support of this system at time \(t = 0.8\) with error \(\varepsilon = 0.1\). For \(m = 1, 2, \ldots\) \(u_m : \Omega \to [0,1]\), for simplicity we assume that if a value of \(u_m\) of a state is not specified, then the value is zero. Below we list the key variables for each \(m\), we present them chronologically from step 7 to 17 of Algorithm 10.

\[
m = 0 \quad u_0(x) = \begin{cases}
0.5, & x = (1,1) \\
0.5, & x = (2,0)
\end{cases} \quad \nabla_0 = \begin{cases}
(1,1) \\
(2,0)
\end{cases} \quad \tau_0 = \varepsilon
\]

\[
m = 1 \quad \nabla_1 = \begin{cases}
(1,1) \\
(0,2) \\
(2,0)
\end{cases} \quad u_1(x) = \begin{cases}
0.179, & x = (2,0) \\
0.128, & x = (0,2)
\end{cases} \quad \nabla_1 = \begin{cases}
(0,2) \\
(2,0)
\end{cases} \quad \tau_1 = 0.1
\]

\[
m = 2 \quad \nabla_2 = \begin{cases}
(1,1) \\
(0,2) \\
(2,0)
\end{cases} \quad u_2(x) = \begin{cases}
0.059, & x = (2,0) \\
0.042, & x = (0,2)
\end{cases} \quad \nabla_2 = \begin{cases}
(1,1) \\
(2,0)
\end{cases} \quad \tau_2 = 0.058
\]

\[
m = 3 \quad \nabla_3 = \begin{cases}
(1,1) \\
(0,2) \\
(2,0)
\end{cases} \quad u_3(x) = \begin{cases}
0.04, & x = (2,0) \\
0.029, & x = (0,2)
\end{cases} \quad \nabla_3 = \begin{cases}
(2,0)
\end{cases} \quad \tau_3 = 0.01
\]

\[
m = 4 \quad \nabla_4 = \begin{cases}
(1,1) \\
(0,2) \\
(2,0)
\end{cases} \quad u_4(x) = \begin{cases}
0.013, & x = (1,1) \\
0.004, & x = (2,0) \\
0.003, & x = (0,2)
\end{cases} \quad \nabla_4 = \begin{cases}
(1,1)
\end{cases} \quad \tau_4 = 0.01
\]

\[
m = 5 \quad \nabla_5 = \begin{cases}
(1,1) \\
(2,0) \\
(0,2)
\end{cases} \quad u_5(x) = \begin{cases}
\end{cases} \quad \nabla_5 = \emptyset \quad \tau_5 < 0.
\]

For this problem the GORDE algorithm will output

\[
\Omega_{\Delta t} = \{(2,0), (1,1), (0,2)\}.
\]
Remark 5.8. The GORDE algorithm is the $N$-step method with an additional feature which gives meaningful weights to new states being added to the domain. A case where the GORDE algorithm would perform less efficiently than the $N$-step method would be if the system had a fast reversible reactions (that is, $A \rightarrow B, B \rightarrow A$). In this case, the GORDE algorithm would keep adding the same states and converge slowly. This case can be avoided in implementation by checking if new states are being added to the domain. In section 5.3, we compute the T-cell Homoeostasis example, which is made up of two reversible reactions. For this example, the GORDE and the $N$-step algorithms added similar amount of states. However, the GORDE algorithm deals with the stiffness of the problem better and computes the approximation 9 times faster than the $N$-step method.

5.2.1 Accuracy

In this section the probability of a particular reaction occurring is deconstructed into simpler components, that is, the probability is determined by summing the probability of travelling along all the paths from the origin to that reaction state. The probabilities of travelling along a path of the reaction state space are derived from the analytical solution of the CME with respect to reaction counts presented in Proposition 5.4. This Section begins with the construction of paths over the reaction space. Then an approximation over the paths is introduced, which is then proved to be an upper bound of the probability of the paths. This approximation is the $u_m$ function in the GORDE algorithm, the section concludes by proving the convergence and accuracy of the GORDE algorithm using the results presented below.

Definition 5.9. We say, $g = (r_0, \ldots, r_m)$, is an admissible path if $r_k \in \Lambda$ for $k = 1, \ldots, m$ and there exists $i_0, \ldots, i_{m-1} \in \{1, \ldots, N_c\}$ such that $r_0 = 0$ and $r_{k+1} = r_k + I_{i_k}$, for $k = 0, \ldots, m - 1$. Here $I_j$ is the identity vector with one in the $j$th position.

We denote $|g| = m$ as the length of the path and $\gamma_m$ as the set of all admissible paths of length $m \in \mathbb{N}$.

By construction, $g$ is a connected path on the reaction counts state space, furthermore, it follows that

$$|g| = \|r_m\|_1.$$

We denote the set of all admissible paths ending in $r \in \Lambda$ by,
\[ S(r) = \{ g = (r_0, \ldots, r_m) \in \gamma_m : r_m = r \}. \]

Since the reaction state space, \( \Lambda \), is a simple positive integer lattice of dimension \( N_r \), we get the relation that

\[ S(r) = \bigcup_{i=1}^{N_r} \{(r_0, \ldots, r_{m-1}, r) : r = r_{m-1} + I_i \text{ with } (r_1, \ldots, r_{m-1}) \in S(r_{m-1})\}. \]

When considering terms, \( r \), on the borders of the positive integer lattice, some of the terms in the union will be empty.

We recall that

\[ \Gamma_{x_0}(r) = x_0 + \mathbb{V}^T r \]

and

\[ \alpha_0(x) = \sum_{j=1}^{N_r} \alpha_j(x) \]

for \( x, x_0 \in \mathbb{N}_0^N \) and \( r \in \mathbb{N}_0^{N_r} \).

**Definition 5.10.** The initial probability of being at \( r \) at time \( t \) and arriving on the path \( g \) is given by

\[ P_g(0, t) := e^{-\alpha_0(\Gamma_{x_0}(0))t}C_g \]  

where \( 0 < C_g \leq 1 \). For \( k = 1, 2, 3, \ldots, |g| \), the probability of being at \( r \) at time \( t \) and arriving on path \( g \) is given by

\[ P_g(k, t) := \alpha_i(\Gamma_{x}(g_{k-1})) \int_0^t e^{-\alpha_0(\Gamma_{x}(g_k))(t-s)}P_g(k-1, s)ds, \]

where \( i \) is the index of the reaction which occurred from \( g_{k-1} \) to \( g_k \).

**Proposition 5.11.** Fix \( t > 0 \). For \( r \in \Lambda \), let \( m = \|r\|_1 \). Let \( g \) and \( \gamma_m \) be the path and the set of all paths as given in Definition 5.9. Let the path probability, denoted by \( P_g(k, t) \), be defined as in Definition 5.10. Define \( S(r) := \{ g \in \gamma_m : g_m = r \} \).

If \( \bar{P}(r, t) \) is given by the integral form in Proposition 5.4 and \( C_g \) in (5.13) is equal to \( C \) in Proposition 5.4, then

\[ \bar{P}(r, t) = \sum_{g \in S(r)} P_g(m, t), \text{ for all } r \in \Lambda. \]
Proof. This result is proven inductively by first showing the result for $m = 1$. Fix $i \in 1, \ldots, N_r$. Proposition 5.4 states

$$\bar{P}(I_i, t) = \alpha_i(\Gamma_x(0)) \int_0^t e^{-\alpha_0(\Gamma_x(I_i))(t-s)} \bar{P}(0, s) ds.$$  \hfill (5.16)

Since $m = 1$, $S(I_i)$ has one element, that is, $S(I_i) = \{(I_i)\}$. Therefore,

$$P(I_i)(1, t) = \alpha_i(\Gamma_x(0)) \int_0^t e^{-\alpha_0(\Gamma_x(I_i))(t-s)} \bar{P}(0, s) ds.$$  \hfill (5.17)

Hence

$$P(I_i, t) = \sum_{g \in S(I_i)} P_g(1, t).$$

Assume that the result is true for $m = k$ where $m \geq 1$. That is for $r \in \Lambda$, where $\|r\|_1 = k$,

$$\bar{P}(r, t) = \sum_{g \in S(r)} P_g(m, t).$$  \hfill (5.18)

Let $r \in \Lambda$ have $\|r\|_1 = k + 1$, then by Proposition 5.4

$$\bar{P}(r_1, t) = \sum_{i=1}^{N_r} \alpha_i(\Gamma_x(r_1 - I_i)) \int_0^t e^{-\alpha_0(\Gamma_x(r_1))(t-s)} \bar{P}(r_1 - I_i, s) ds$$  \hfill (5.19)

$$= \sum_{i=1}^{N_r} \alpha_i(\Gamma_x(r_1 - I_i)) \int_0^t e^{-\alpha_0(\Gamma_x(r_1))(t-s)} \sum_{g \in S(r_1 - I_i)} P_g(m, s) ds,$$  \hfill (5.20)

by our inductive step,

$$= \sum_{i=1}^{N_r} \sum_{g \in S(r_1 - I_i)} \alpha_i(\Gamma_x(r_1 - I_i)) \int_0^t e^{-\alpha_0(\Gamma_x(r_1))(t-s)} P_g(m, s) ds.$$  \hfill (5.21)

Let $i, j \in 1, \ldots, N_r$, define the map $\psi_j : \gamma_m \mapsto \gamma_{m+1}$ by $\psi_j(g) := (g_1, \ldots, g_m, g_m + I_j)$.

Let $H(r_1 - I_i) := \{\psi_j(g) : g \in S(r - I_i) and j = 1, \ldots N_r\}$. Since $S(r - I_j) \cap S(r - I_i) = \emptyset$ for $i \neq j$, it follows that

$$H(r_1 - I_i) \cap H(r_1 - I_j) = \emptyset for i \neq j.$$

Then $S(r_1) = \bigcap_{i=1}^{N_r} H(r_1 - I_i)$. As a result
\[ \bar{P}(r_1, t) = \sum_{i=1}^{N_r} \sum_{g \in S(r_1 - I_i)} \alpha_i(\Gamma_x(r_1 - I_i)) \int_0^t e^{-\alpha_0(\Gamma_x(r_1 - I_i))(t-s)} P_g(m, s) ds \]
\[ = \sum_{i=1}^{N_r} \sum_{g \in H(r_1 - I_i)} P_g(m + 1, t) \]
\[ = \sum_{g \in S(r_1)} P_g(m + 1, t). \]

Hence by mathematical induction, for any \( r \in \Lambda \),
\[ \bar{P}(r, t) = \sum_{g \in S(r)} P_g(|g|, t). \]

The Proposition above reduces the problem of computing states in the reaction space to computing independent paths through the reaction state space, Figure 5.2. Results about probabilities of paths on the reaction spaces are provided below. These are used in convergence and accuracy results of the GORDE algorithm.

Consider a decomposition of the reaction space into non-intersecting subsets based on number of reactions fired. For simplicity, in the proofs, we redefine for \( i = 1, \ldots, N_r \) \( R_i(r) := r + I_i \), where \( I_i \) is the unit vector with 1 in the \( i \)th position.
Definition 5.12. Define a partitioning of $\Lambda$ by

$$\nabla_1 := \bigcup_{i=1}^{N_r} R_i(0), \quad (5.23)$$

and for $m = 2, 3, \ldots$,

$$\nabla_m := \bigcup_{i=1}^{N_r} R_i(\nabla_{m-1}). \quad (5.24)$$

The definition above is also equivalent to

$$\nabla_m := \{ r \in \Lambda : |r| = m \}. \quad (5.25)$$

The following lemma states that if an admissible path gated at some finite step, then the probability that accumulates at the gate forms an upper bound of the sum of the real probability of all the steps past the gate.

Lemma 5.13. Fix $x \in \Omega$. Let $\Lambda = \mathbb{N}^N$ be the state space of the reaction counts. Define $\Gamma_x(r) = x + \nabla^T r$ for all $r \in \Lambda$. For $m = 1, 2, 3, \ldots$, let $\nabla_m$ be as defined in (5.25). Also, for $0 = (0, \ldots, 0) \in \Lambda$, let

$$\bar{P}(0, t) = e^{-\alpha_0(\Gamma_x(0))t} C \quad (5.26)$$

where $C \in (0, 1]$. Furthermore for $r \in \Lambda$, let

$$\bar{P}(r, t) = \sum_{i=1}^{N_r} \alpha_i(\Gamma_x(r - I_i)) \int_0^t e^{-\alpha_0(\Gamma_x(r))(t-s)} \bar{P}(r - I_i, s) ds. \quad (5.27)$$

Then for $m = 1, 2, \ldots$,

$$\sum_{r_1 \in \nabla_m} \sum_{i=1}^{N_r} \alpha_i(\Gamma_x(r_1)) \int_0^{\Delta t} \bar{P}(r_1, s) ds \geq \sum_{n=m}^{\infty} \sum_{r_2 \in \nabla_n} \bar{P}(r_2, \Delta t). \quad (5.28)$$

Proof. Fix $m \in \mathbb{N}$. Truncate all the states in $\Lambda$ whose coordinates sum is more than $m + 1$ and set the outgoing propensities of the states in $\nabla_{m+1}$ to zero. This gives a finite dimensional CME problem and the finite state projection approximation for states up to $\nabla_m$ are the true probabilities (c.f. Proposition 5.4). The sum over all probabilities at time $\Delta t > 0$ is

$$P(0, \Delta t) + \sum_{r_1 \in \nabla_1} P(r_1; \Delta t) + \ldots + \sum_{r_m \in \nabla_m} P(r_m; \Delta t) + \sum_{r_m \in \nabla_m} \sum_{i=1}^{N_r} \alpha_i(\Gamma_x(r_m)) \int_0^{\Delta t} P(r_m, s) ds \quad (5.29)$$

Since the problem is finite dimensional, we conserve all our probability by Theorem 2.11 and
5.2. GORDE ALGORITHM

\[
\left( P(0, \Delta t) + \sum_{r_1 \in V_1} P(r_1, \Delta t) + \ldots + \sum_{r_m \in V_m} P(r_m, \Delta t) + \sum_{r_m \in V_m} \sum_{i=1}^{N_r} \alpha_i(\Gamma_x(r_m)) \int_0^{\Delta t} P(r_m, s)ds \right) = C
\]

(5.30)

However, for the infinite system, Theorem 2.11 states

\[
\bar{P}(0, \Delta t) + \sum_{m=1}^{\infty} \sum_{r_m \in V_m} \bar{P}(r_m, \Delta t) \leq C.
\]

(5.31)

Substituting (5.30) into (5.31) for \( C \) and removing like terms gives

\[
\sum_{r_1 \in V_m} \sum_{i=1}^{N_r} \alpha_i(\Gamma_x(r_1)) \int_0^{\Delta t} \bar{P}(r_1, s)ds \geq \sum_{n=m}^{\infty} \sum_{r_2 \in V_n} \bar{P}(r_2, \Delta t).
\]

The following lemma states that gating at the second step of an admissible path, gives an upper bound for the true probability at the second step of the path.

**Lemma 5.14.** For \( 0 < C \leq 1 \), \( 0 < \beta_1 \leq \alpha_1 \), \( \alpha_2 > 0 \) and \( t > 0 \), let

\[
f_t(1) := e^{-\alpha_1 t} C,
\]

(5.32)

and

\[
f_t(2) := \beta_1 \int_0^t e^{-\alpha_2 (t-s)} f_s(1)ds.
\]

(5.33)

If for \( \Delta t > 0 \)

\[
u_{\Delta t}(2) := \beta_1 \int_0^{\Delta t} f_s(1)ds
\]

(5.34)

and

\[
v_t(2) := e^{-\alpha_2 t} u_{\Delta t}(2), \text{ for } t \in [0, \Delta t],
\]

(5.35)

then

\[
\int_0^{\Delta t} f_s(2)ds \leq \int_0^{\Delta t} v_s(2)ds.
\]

(5.36)

and

\[
f_{\Delta t}(2) \leq u_{\Delta t}(2).
\]

(5.37)
Proof. We have
\begin{align*}
f_t(2) &= \frac{C\beta_1}{\alpha_1 - \alpha_2} (e^{-\alpha_2 t} - e^{-\alpha_1 t}), \quad (5.38) \\
u_{\Delta t}(2) &= \frac{C\beta_1}{\alpha_1} (1 - e^{-\alpha_1 \Delta t}), \quad (5.39) \\
v_t(2) &= \frac{C\beta_1}{\alpha_1} e^{-\alpha_2 t} (1 - e^{-\alpha_1 \Delta t}). \quad (5.40)
\end{align*}

Also
\begin{align*}
\int_0^{\Delta t} f_s(2) ds &= \frac{C\beta_1}{(\alpha_1 - \alpha_2)\alpha_1 \alpha_2} (\alpha_1 - \alpha_2 + \alpha_2 e^{-\alpha_1 \Delta t} - \alpha_1 e^{-\alpha_2 \Delta t}), \quad (5.41) \\
\text{and} \\
\int_0^{\Delta t} v_s(2) ds &= \frac{C\beta_1}{\alpha_1 \alpha_2} (1 - e^{-\alpha_1 \Delta t}) (1 - e^{-\alpha_2 \Delta t}). \quad (5.42)
\end{align*}

Subtracting \( \int_0^{\Delta t} f_s(2) ds \) from \( \int_0^{\Delta t} v_s(2) ds \) and eliminating the like terms, we have

\begin{align*}
\int_0^{\Delta t} v_s(2) ds - \int_0^{\Delta t} f_s(2) ds &= \frac{\beta_0 C (e^{-\alpha_2 \Delta t} \alpha_2 - e^{-\alpha_1 \Delta t} \alpha_1 + (\alpha_1 - \alpha_2) e^{-(\alpha_1 + \alpha_2) \Delta t})}{(\alpha_1 - \alpha_2)\alpha_2 \alpha_1}, \\
(5.43)
\end{align*}

If we define
\begin{align*}
\Psi(a, b) := \frac{\beta_0 C (e^{-b \Delta t} b - e^{-a \Delta t} a + (a - b) e^{-(a+b) \Delta t})}{(a - b)ab}, \quad (5.44)
\end{align*}

then
\begin{align*}
\Psi(\alpha_1, \alpha_2) = \Psi(\alpha_2, \alpha_1).
\end{align*}

Hence only one case of \( \alpha_1 \) and \( \alpha_2 \) is considered.

Assume \( \alpha_1 < \alpha_2 \). Then

\begin{align*}
\int_0^{\Delta t} v_s(2) ds - \int_0^{\Delta t} f_s(2) ds &\geq \frac{\beta_0 C ((e^{-\alpha_2 \Delta t} - e^{-\alpha_1 \Delta t}) \alpha_2 + (\alpha_1 - \alpha_2) e^{-(\alpha_1 + \alpha_2) \Delta t})}{(\alpha_1 - \alpha_2)\alpha_2 \alpha_1}, \\
&\geq 0,
\end{align*}

Therefore
\begin{align*}
\int_0^{\Delta t} f_s(2) ds &\leq \int_0^{\Delta t} v_s(2) ds.
\end{align*}

To show (5.37), we have
5.2. GORDE ALGORITHM

\[ f_{\Delta t}(2) := \beta_1 \int_0^{\Delta t} e^{-\alpha_2(\Delta t-s)} f_s(1) ds, \]
\[ \leq \beta_1 \int_0^{\Delta t} f_s(1) ds, \]
\[ = u_{\Delta t}(2). \]

We now introduce the notion of un-gating. Un-gating is when we reset the clock and set the probability that has accumulated in the gated state as the initial value at the gate and propagate probability forwards through the path. The lemma below states that the approximation made by gating and un-gating gives an upper bound for the real probability distribution over the path.

**Lemma 5.15.** For a constant \( 0 \leq C_f \leq 1 \), constants \( 0 < \beta_k \leq \alpha_k \), where \( k = 1, 2, \ldots \), and \( t \in [0, \Delta t] \), we define

\[ f_t(1) := e^{-\alpha_1 t} C_f \]  (5.45)

and

\[ f_t(k) := \beta_{k-1} \int_0^t e^{-\alpha_k (t-s)} f_s(k-1) ds \]  (5.46)

for \( k = 2, 3, \ldots \).

Also for \( 0 \leq C_g \leq 1 \), let

\[ g_t(2) := e^{-\alpha_2 t} C_g \]  (5.47)

and

\[ g_t(k) := \beta_{k-1} \int_0^t e^{-\alpha_k (t-s)} g_s(k-1) ds \]  (5.48)

for \( k = 3, 4, \ldots \).

For \( \Delta t > 0 \), if

\[ \int_0^{\Delta t} f_s(2) ds \leq \int_0^{\Delta t} g_s(2) ds, \]  (5.49)

then for all \( k = 3, 4, \ldots \) and \( t \in [0, \Delta t] \),

\[ f_t(k) \leq g_t(k), \]  (5.50)

and

\[ \int_0^{\Delta t} f_s(k) ds \leq \int_0^{\Delta t} g_s(k) ds. \]  (5.51)
Proof. This result is proven inductively by first showing it holds for $k = 3$. For any $t \in [0, \Delta t]$, 
\[
    f_t(3) - g_t(3) = \beta_2 \int_0^t e^{-\alpha_3(t-s)}(f_s(2) - g_s(2))ds 
    \leq \beta_2 \int_0^t (f_s(2) - g_s(2))ds 
    \leq \beta_2 \int_0^{\Delta t} (f_s(2) - g_s(2))ds 
    \leq 0, \text{ by our assumption (5.49)}.
\]

Assume the result to be true for $k = m$ where $m \geq 3$. That is for any $t \in [0, \Delta t]$, 
\[
    f_t(m) \leq g_t(m). \quad (5.52)
\]

Consider the case $k = m + 1$. For any $t \in [0, \Delta t]$, 
\[
    f_t(m + 1) - g_t(m + 1) = \beta_m \int_0^t e^{\alpha_{m+1}(t-s)}(f_s(m) - g_s(m))ds 
    \leq \beta_m \int_0^t (f_s(m) - g_s(m))ds 
    \leq 0, \text{ by our assumption (5.52)}.
\]

Therefore by mathematical induction, for $t \in [0, \Delta t]$, 
\[
    f_t(k) \leq g_t(k),
\]
for all $k = 3, 4, \ldots$. Then
\[
    \int_0^{\Delta t} f_s(k)ds \leq \int_0^{\Delta t} g_s(k)ds.
\]

The following lemma demonstrates that we can inductively gate and un-gate along an admissible path and that at each gate, we will have an upper bound to the real probability of being at that gate.

Lemma 5.16. For a constant $0 \leq C_v \leq 1$, constants $0 < \beta_k \leq \alpha_k$, where $k = 1, 2, \ldots$, and $t \in [0, \Delta t]$, we define
\[
    v_t(1, 0) := e^{-\alpha_1 t}C_v \quad (5.53)
\]
5.2. GORDE ALGORITHM

and

\[ v_t(1, j) := \beta_{j-1} \int_0^t e^{-\alpha_k(t-s)} v_s(0, j - 1) ds \]  

(5.54)

for \( j = 1, 2, \ldots \).

Furthermore, define

\[ u_{\Delta t}(2) := \beta_{k-1} \int_0^{\Delta t} v_s(1, 0) ds, \]  

(5.55)

and for \( k = 3, 4, \ldots \), define

\[ u_{\Delta t}(k) := \beta_{k-1} \int_0^{\Delta t} v_s(k - 1, 0) ds, \]  

(5.56)

where for \( k = 2, 3, \ldots \),

\[ v_t(k, 0) := e^{-\alpha_k t} u_{\Delta t}(k), \quad t \in [0, \Delta t], \]  

(5.57)

and for \( j = 1, 2, \ldots \),

\[ v_t(k, j) := \beta_{k-1+j} \int_0^t e^{-\alpha_k+j(t-s)} v_s(k, j - 1) ds. \]  

(5.58)

Then for all \( k = 2, 3, \ldots \),

\[ v_{\Delta t}(1, k) \leq u_{\Delta t}(k), \]  

(5.59)

and

\[ \beta_{k-1} \int_0^{\Delta t} v_s(1, k - 1) ds \leq u_{\Delta t}(k). \]  

(5.60)

Proof. Fix \( t \in [0, \Delta t] \). For \( k = 2, 3, \ldots \), We first prove the following claim inductively,

\[ \text{claim}(k) := \begin{cases} u_{\Delta t}(k) \geq v_{\Delta t}(k - 1, 1) & \text{for } j = 1 \\ v_{\Delta t}(k - 1, j + 1) \leq v_{\Delta t}(k, j) & \text{for } j > 1. \end{cases} \]

We start with our base case and prove the result for \( k = 2 \).

Applying Lemma 5.14 to \( v_t(1, \cdot) \) we have

\[ v_{\Delta t}(1, 1) \leq u_{\Delta t}(2) \]  

(5.61)

and

\[ \int_0^{\Delta t} v_s(1, 1) \leq \int_0^{\Delta t} v_s(2, 0) ds. \]
Applying Lemma 5.15 with the above integral inequality, for all \( j = 2, 3, \ldots \) we have

\[
v_{\Delta t}(1, j) \leq v_{\Delta t}(2, j - 1),
\]

(5.62)

and

\[
\int_0^{\Delta t} v_s(1, j) ds \leq \int_0^{\Delta t} v_s(2, j - 1) ds.
\]

(5.63)

Equation (5.61) and (5.62) prove our base claim. For the inductive step, we assume for \( k = m \), where \( m \geq 2 \), the claim is true, that is,

\[
\text{claim}(m) := \begin{cases} 
  u_{\Delta t}(m) \geq v_{\Delta t}(m - 1, 1) & \text{for } j = 1 \\
  v_{\Delta t}(m - 1, j + 1) \leq v_{\Delta t}(m, j) & \text{for } j > 1.
\end{cases}
\]

We prove the result for \( k = m + 1 \) by considering the two cases in the claim.

Case \( j = 1 \): We show \( u_{\Delta t}(m + 1) \geq v_{\Delta t}(m, 1) \), first. We have

\[
u_{\Delta t}(m + 1) := \beta_m \int_0^{\Delta t} v_s(m, 1) ds,
\]

(5.64)

with

\[
v_t(m, 1) := \beta_m \int_0^t e^{-\alpha_{m+1}(t-s)} v_s(m, 0) ds,
\]

(5.65)

and

\[
v_t(m, 0) := \int_0^t e^{-\alpha_m s} u_{\Delta t}(m) ds.
\]

(5.66)

Apply Lemma 5.14 to \( v_t(m, 0) \), then

\[
u_{\Delta t}(m + 1) \geq v_{\Delta t}(m, 1).
\]

(5.67)

and

\[
\int_0^{\Delta t} v_s(m, 1) ds \leq \int_0^{\Delta t} v_s(m + 1, 0) ds.
\]

(5.68)

Hence for the case \( j = 1 \), \( \text{claim}(m + 1) \) follows from \( \text{claim}(m) \).

Case \( j > 1 \): Applying Lemma 5.15, that is, \( f_t(1) = v_t(m, 0), g_t(2) = v_t(m + 1, 0) \) and (5.68), then for \( j = 1, 2, \ldots \), we have

\[
v_{\Delta t}(m, j + 1) \leq v_{\Delta t}(m + 1, j),
\]
and
\[ \int_0^{\Delta t} v_s(m, j + 1)ds \leq \int_0^{\Delta t} v_s(m + 1, j)ds. \quad (5.69) \]

Therefore by mathematical induction, for \( k = 1, 2, \ldots \)
\[ u_{\Delta t}(k) \geq v_{\Delta t}(k - 1, 1), \quad \text{for } j = 1, \quad (5.70) \]
and
\[ v_{\Delta t}(k - 1, j + 1) \leq v_{\Delta t}(k, j), \quad \text{for } j > 1. \quad (5.71) \]

Then
\[
\begin{align*}
    u_{\Delta t}(k) &\geq v_{\Delta t}(k - 1, 1), \quad \text{by (5.70)} \\
    &\geq v_{\Delta t}(k - 2, 2), \quad \text{by (5.71)} \\
    \vdots \\
    &\geq v_{\Delta t}(2, k - 2) \\
    &\geq v_{\Delta t}(1, k - 1), \quad \text{by (5.62)}. \\
\end{align*}
\]

For the second condition, recursively apply (5.69) to obtain
\[
\begin{align*}
    u_{\Delta t}(k) := \beta_{k-1} \int_0^{\Delta t} v_s(k - 1, 0)ds \\
    &\geq \beta_{k-1} \int_0^{\Delta t} v_s(k - 2, 1)ds, \quad \text{by (5.69)} \\
    \vdots \\
    &\geq \beta_{k-1} \int_0^{\Delta t} v_s(2, k - 3)ds, \quad \text{by (5.63)} \\
    &\geq \beta_{k-1} \int_0^{\Delta t} v_s(1, k - 2)ds.
\end{align*}
\]

**Theorem 5.17.** Let \( x \in \Omega \) be the starting point of the CME problem with probability \( w \in (0, 1] \). Define \( \tilde{w} \) be a vector indexed by \( \Omega \), where it takes value \( w \) at \( x \) and zero elsewhere. Let \( \lambda \) be the reaction state space with \( N_r \) reactions and \( \mathbb{V} \) be the stoichiometric matrix. For \( r \in \Lambda \), let \( \Gamma_x(r) := x + \mathbb{V}^T r \) be the affine mapping between the state spaces. For \( m = 1, 2, \ldots \), let \( \gamma_m \) be the set of all paths of length \( m \) from the origin, as defined in Definition 5.9. Also, let the partition be \( \nabla_m \) as defined in Definition 5.12. Let the probability of being on point \( 0 \) be
\[ P_g(0, t) := e^{-\alpha_0(\Gamma_x(0))t}w \]  

(5.72)

and the probability for \( k \in 1, \ldots, m \) be

\[ P_g(k, t) := \alpha_i(\Gamma_x(g_{k-1})) \int_0^t e^{-\alpha_0(\Gamma_x(g_k))(t-s)} P_g(k - 1, s) ds, \]  

(5.73)

where \( i \) is the reaction which moves \( g_{k-1} \) to \( g_k \).

Define

\[ u_g(1, \Delta t) := \alpha_j(\Gamma_x(0)) \int_0^{\Delta t} P_g(0, s) ds, \]  

(5.74)

and for \( k = 2, 3, \ldots, \)

\[ u_g(k, \Delta t) := \alpha_i(\Gamma_x(g_{k-1})) u_g(k - 1, \Delta t) \int_0^{\Delta t} e^{-\alpha_0(\Gamma_x(g_{k-1}))(s)} ds, \]  

(5.75)

where \( i \) is the reaction which maps \( g_{k-1} \) to \( g_k \) and \( j \) is the reaction which maps \( 0 \) to \( g_1 \).

Define

\[ \Omega_{m, \Delta t} := \bigcup_{n=1}^m \Gamma_x(\nabla_n) \text{ and } S(r) := \{ g \in \gamma_m : g_m = r \}. \]

If for \( m \geq 1 \) and \( \varepsilon > 0 \),

\[ \sum_{r \in \nabla_m} \sum_{g \in S(r)} u_g(m, \Delta t) \leq \varepsilon, \]  

(5.76)

then

\[ \| e^{A\Delta t} \tilde{w} - w_{\Delta t} \|_1 \leq \varepsilon, \]  

(5.77)

where \( w_{\Delta t} \) is the FSP of \( \tilde{w} \) for a time step \( \Delta t \) on the state space \( \Omega_{m, \Delta t} \).

**Proof.** Fix \( m \in \mathbb{N} \) and \( r \in \nabla_m \).

For \( i, j \in \{1, \ldots, N_r\} \) let

\[ S_i(r) := \{ g \in \gamma_m : g_{m-1} = r - I_i \}. \]  

(5.78)

Then

\[ S_i(r) \cap S_j(r) = \emptyset \text{ for } i \neq j. \]  

(5.79)
5.2. GORDE ALGORITHM

Therefore

\[ S(r) := \bigcup_{i=1}^{N_r} S_i(r) = \{ g \in \gamma_m : g_m = r \}. \]  

Fix \( i \in \{1, \ldots, N_r\} \). We have

\[ \hat{P}(r - I_i, t) = \sum_{g \in S_i(r)} P_g(m - 1, t), \]

\[ \alpha_i(\Gamma_x(r - I_i)) \hat{P}(r - I_i, t) = \sum_{g \in S_i(r)} \alpha_i(\Gamma_x(r - I_i)) P_g(m - 1, t), \]

hence

\[ \int_0^{\Delta t} \alpha_i(\Gamma_x(r - I_i)) \hat{P}(r - I_i, s) ds = \int_0^{\Delta t} \sum_{g \in S_i(r)} \alpha_i(\Gamma_x(r - I_i)) P_g(m - 1, s) ds \]

\[ \leq \sum_{g \in S_i(r)} u_g(m, \Delta t), \]  by Lemma 5.16.

Summing over all \( i = 1, \ldots, N_r \) we obtain

\[ \sum_{i=1}^{N_r} \int_0^{\Delta t} \alpha_i(\Gamma_x(r - I_i)) \hat{P}(r - I_i, s) ds \leq \sum_{i=1}^{N_r} \sum_{g \in S_i(r)} u_g(m, \Delta t), \]

\[ = \sum_{g \in S(r)} u_g(m, \Delta t). \]

Summing over all \( r \in \nabla_m \) we obtain

\[ \sum_{r \in \nabla_m} \sum_{i=1}^{N_r} \int_0^{\Delta t} \alpha_i(\Gamma_x(r - I_i)) \hat{P}(r - I_i, s) ds \leq \sum_{r \in \nabla_m} \sum_{g \in S(r)} u_g(m, \Delta t). \] (5.81)

Applying Lemma 5.13 to the right hand size of (5.81), we get

\[ \sum_{n=m}^{\infty} \sum_{r \in \nabla_n} \hat{P}(r, \Delta t) \leq \sum_{r \in \nabla_m} \sum_{i=1}^{N_r} \int_0^{\Delta t} \alpha_i(\Gamma_x(r - I_i)) \hat{P}(r - I_i, s) ds \]

\[ \leq \sum_{r \in \nabla_m} \sum_{g \in S(r)} u_g(m, \Delta t) \]  by assumption (5.76).
Then

$$\left\| \vec{p}_{\Delta t} - \vec{p}_{\Delta t} |_{\bigcup_{n=0}^{m} \nabla_n} \right\|_1 = \sum_{n=m}^\infty \sum_{r \in \nabla_n} \vec{p}(r, \Delta t),$$

$$\leq \sum_{r \in \nabla_m} \sum_{g \in S(r)} u_g(m, \Delta t), \quad \text{by (5.82),}$$

$$\leq \varepsilon, \quad \text{by assumption (5.76).}$$

Let \( w_{\Delta t} \) be the mapping forward of \( \vec{p}_{\Delta t} |_{\bigcup_{n=0}^{m} \nabla_n} \) by the affine map \( \Gamma_x \). Then

$$\left\| e^{A_{\Delta t}} \vec{w} - w_{\Delta t} \right\|_1 = \left\| \vec{p}_{\Delta t} - \vec{p}_{\Delta t} |_{\bigcup_{n=0}^{m} \nabla_n} \right\|_1$$

$$\leq \varepsilon.$$

\[ \square \]

**Corollary 5.18.** Let \( p \in \ell_1 \) with \( \Omega \) being its support, \( p(x) \in (0, 1] \) for \( x \in \Omega \), and \( \|p\|_1 \leq 1 \).

Let \( \Delta t > 0 \) and define \( w_x \) to be a vector with support at only \( x \), that is \( w_x(x) = p(x) \) and \( w_x(y) = 0 \) for \( y \in \Omega \setminus \{x\} \).

Let \( \Omega_{\Delta t} \) be given by the GORDE Algorithm 10, with parameters as defined in Table 5.1 and the initial vector set to \( w_x \) and error \( \varepsilon/|\Omega| \).

If \( \vec{w}_{x,\Delta t} \) is the FSP approximation of the CME problem for time \( \Delta t \) over the domain \( \Omega_{\Delta t} \), then

$$\left\| e^{A_{\Delta t}} p - \sum_{x \in \Omega} \vec{w}_{x,\Delta t} \right\|_1 \leq \varepsilon.$$

**Proof.** Fix \( x \in \Omega \). Let \( w_x \) be the initial condition for the CME problem with respect to reaction counts. Let \( A_x \) be the infinitesimal operator for CME problem with respect to species. By definition we have,

$$\sum_{x \in \Omega} e^{A_{\Delta t}} w_x = e^{A_{\Delta t}} p. \quad (5.83)$$

Applying Theorem 5.17

$$\left\| e^{A_{\Delta t}} w_x - \vec{w}_{x,\Delta t} \right\|_1 \leq \frac{\varepsilon}{|\Omega|}. \quad (5.84)$$
Therefore

\[
\left\| e^{A_s \Delta t} p - \sum_{x \in \Omega} \tilde{w}_x \Delta t \right\|_1 = \left\| \sum_{x \in \Omega} e^{A_s \Delta t} w_x - \tilde{w}_x \Delta t \right\|_1, \quad \text{by (5.83)},
\]

\[
\leq \sum_{x \in \Omega} \left\| e^{A_s \Delta t} w_x - \tilde{w}_x \Delta t \right\|_1,
\]

\[
\leq \sum_{x \in \Omega} \frac{\varepsilon}{|\Omega|}, \quad \text{by (5.84)}
\]

\[
= \varepsilon.
\]

\[
\square
\]

Corollary 5.18 guarantees that the GORDE algorithm gives the right support with respect to the error we subscribe for the future compute of the CME problem.

### 5.2.2 Convergence

**Proposition 5.19.** Let \( x \in \Omega \) be the starting point of the CME problem with probability \( w_x \in (0, 1] \). Let \( \Lambda \) be the reaction state space with \( N_r \) reactions and \( V \) be the stoichiometric matrix. For \( r \in \Lambda \), let \( \Gamma_x(r) := x + V^T r \) be the affine mapping between the state spaces. For \( m = 1, 2, \ldots \), let \( \gamma_m \) be the set of all paths of length \( m \) from the origin, as defined in Definition 5.9. Also, let the partition \( \nabla_m \) be defined as in Definition 5.12. Fix \( \Delta t > 0 \).

Define

\[
u_g(1, \Delta t) := \alpha_j(\Gamma_x(0)) \int_0^{\Delta t} P_g(0, s) ds,
\]

(5.85)

and for \( k = 2, 3, \ldots \) define

\[
u_g(k, \Delta t) := \alpha_i(\Gamma_x(g_{k-1})) \nu_g(k-1, \Delta t) \int_0^{\Delta t} e^{-\alpha_0(\Gamma_x(g_{k-1}))s} ds,
\]

(5.86)

where \( i \) is the reaction mapping \( g_{k-1} \) to \( g_k \) and \( j \) is the reaction mapping \( 0 \) to \( g_1 \).

Define \( \psi_i : \gamma_m \mapsto \gamma_{m+1} \) by \( \psi_i(g) := (g_1, \ldots, g_m, g_m + I_i) \). For \( \Delta t > 0 \), given \( 0 < \varepsilon < w_x \), there exists \( m \in \mathbb{N} \) such that

\[
\sum_{g \in \gamma_m} u_g(m, \Delta t) \leq \varepsilon.
\]
Proof. Fix $m \in \mathbb{N}$. For $g \in \gamma_m$

\[ u_g(m, \Delta t) = e^{-\alpha_0(\Gamma_x(g_{m-1}))\Delta t}u_g(m, \Delta t) + \sum_{i=1}^{N_r} \alpha_i(\Gamma_x(g_{m-1})) (1 - e^{-\alpha_0(\Gamma_x(g_{m-1}))\Delta t})u_g(m, \Delta t), \]

\[ = e^{-\alpha_0(\Gamma_x(g_{m-1}))\Delta t}u_g(m, \Delta t) + \sum_{i=1}^{N_r} \alpha_i(\Gamma_x(g_{m-1})) u_g(m, \Delta t) \int_{0}^{\Delta t} e^{-\alpha_0(\Gamma_x(g_{m-1}))s} ds, \]

\[ = e^{-\alpha_0(\Gamma_x(g_{m-1}))\Delta t}u_g(m, \Delta t) + \sum_{i=1}^{N_r} u_{\psi_i}(g)(m + 1, \Delta t). \] (5.87)

Then

\[ u_g(m, \Delta t) > \sum_{i=1}^{N_r} u_{\psi_i}(g)(m + 1, \Delta t). \]

Summing over all $g \in \gamma_m$,

\[ \sum_{g \in \gamma_m} u_g(m, \Delta t) > \sum_{g \in \gamma_m} \sum_{i=1}^{N_r} u_{\psi_i}(g)(m + 1, \Delta t). \]

The set \( \{\psi_i(g) : g \in \gamma_m \text{ and } i = 1, \ldots, N_r\} = \gamma_{m+1} \). Hence, that above inequality reduces to

\[ \sum_{g \in \gamma_m} u_g(m, \Delta t) > \sum_{h \in \gamma_{m+1}} u_h(m + 1, \Delta t). \] (5.88)

The sequence \( (\sum_{g \in \gamma_m} u_g(m, \Delta t))_{m=1}^{\infty} \) is strictly decreasing and non-negative. Therefore the \( \lim_{n \to \infty} \sum_{g \in \gamma_m} u_g(m, \Delta t) = 0 \). Since the sequence is bounded above by \( w_x \), for \( 0 < \varepsilon < w_x \), there exists a point \( n \) such that

\[ \sum_{g \in \gamma_n} u_g(n, \Delta t) \leq \varepsilon. \]

\[ \square \]

Remark 5.20. There is a very simple example where the GORDE algorithm and any other domain finding method fails. Let us consider a continuous time Markov process on the integers \( \{0, 1, 2, 3, \ldots\} \) with propensity \( 2^n \) of transitioning from state \( n \) to \( n + 1 \). Let the initial state be zero. Then, after a very small time step, \( (\ll 0.1) \), the system behaves naturally. However, soon after, the \( 2^n \) propensity pushes the stochastic process really quickly towards infinity. This behaviour is referred to as an explosion [15]. It is very difficult to construct a probability distribution for this particular problem. However, for simple propensities which
arise from mass action kinetics \cite{35}, we have finite expectations and unique probability distributions which make the stochastic processes well-behaved enough to construct approximations.

5.3 Numerical Experiments

To demonstrate the effectiveness of the GORDE method, we compute the Göutsias’ model \cite{3.3.1}, the Goldbeter-Koshland Switch \cite{3.3.2} and the T-cell Homoeostasis model from Example 2.6. We compare the domain for the following implementations: FSP with GORDE (Algorithm 10), FSP with N-step algorithm (Algorithm 2) and OFSP with N-step algorithm. The N-step domain expander is configured to $N = 1$. Both the N-step and the GORDE algorithms are written to be called on by CMEPy to nominate new states to add to the state space when a time step loses more error than prescribed.

The implementations were run on a Quad-Core AMD Opteron(tm) Processor 8360 SE with 4GB of RAM.

<table>
<thead>
<tr>
<th>$t_{final} = 300$</th>
<th>FSP N-step</th>
<th>FSP GORDE</th>
<th>OFSP N-step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (s)</td>
<td>1036</td>
<td>259</td>
<td>116</td>
</tr>
<tr>
<td>Domain Size</td>
<td>110931</td>
<td>30646</td>
<td>22375</td>
</tr>
<tr>
<td>Error</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
</tr>
</tbody>
</table>

Table 5.2: GORDE and N-step Algorithm comparisons for the Göutsias’ model (§3.3.1)

<table>
<thead>
<tr>
<th>$t_{final} = 20$</th>
<th>FSP N-step</th>
<th>FSP GORDE</th>
<th>OFSP N-step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (s)</td>
<td>10900</td>
<td>1599</td>
<td>214</td>
</tr>
<tr>
<td>Domain Size</td>
<td>171006</td>
<td>27588</td>
<td>4793</td>
</tr>
<tr>
<td>Error</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
</tr>
</tbody>
</table>

Table 5.3: GORDE and N-step Algorithm comparisons for G-K Switch (§3.3.2).

For the T-cell model, there was stiffness within a time unit of one. Hence, a time step of $\Delta t = 0.01$ had to be taken. However, the GORDE method performed up to a time step of $\Delta t = 0.1$. 
Table 5.4: *GORDE* and *N*-step Algorithm comparisons for T-Cell Homoeostasis (Example 2.6).

We see in Table 5.3 and Table 5.2 that the GORDE algorithm performs far superior to the *N*-step method for the FSP implementation. This is because the GORDE method constructs the different reactions paths which the system can undergo and applies a metric to retain the most important paths. The addition of the metric makes the domain grow in the direction that the support grows in. However, with the *N*-step method, the domain grows a step in each direction, calculating improbable reactions and storing them unnecessarily. The GORDE algorithm was applied to the FSP method only, however Table 5.3 and Table 5.2 show that the support of the GORDE algorithm is close to an OFSP method. While OFSP can be implemented with the GORDE Algorithm, the conventional method of simply truncating states after completing a time step is an inefficient exercise when performing domain selection with the GORDE Algorithm. That is, the GORDE algorithm implicitly evolves the most possible paths. Truncating post step removes future paths which have been selected by the GORDE algorithm. The GORDE algorithm selects states which will be useful in the future. These states, in the current time steps, have a small probability and are hence
5.4 Discussion

The GORDE algorithm is the first domain selection algorithm which has been mathematically proven to return the support which the prescribed error. Methods such as the $N$-step and the SW have shown that while the method converges, the only way to know if the right domain is found is by constructing an approximation truncated. The GORDE algorithm would thus have to reconstruct these paths resulting in a longer computation time. Hence, in order to apply the GORDE algorithm to the OFSP method, the compression must be incorporated into the GORDE method itself. This is discussed further below.
and observing the lost error. If too much error is lost, the method adds more
states. We have shown that the GORDE algorithm is accurate and convergent.
Furthermore, in the numerical experiments, the GORDE algorithm for the FSP
method is 4 to 10 times faster. More importantly, the domain size for the GORDE
algorithm is close to that of the OFSP $N$-step method, which is known to be order
optimal. This implies that the GORDE algorithm selects the most relevant future
states.

The GORDE algorithm only adds states and does not remove them. While
the addition of a step to remove redundant states would result in a speed-up, a
truncation after a number of steps is not a logical choice. Hence, an extra step
needs to be added to the GORDE algorithm to remove parts of the state space
that the probability is moving away from. This extra step should be implemented
to ensure that the resulting support can be proven to be of optimal order.
Chapter 6

Conclusion

In this thesis, we proved that regardless of the dimension, approximation of the CME via an FSP type method is simpler than a stochastic simulation method when the minimal support is known. In this thesis, we have shown that the number of realisations needed will always be larger than the cardinality of the minimal support. This point gives us a strong motivation to study domain finding/constructing methods. By estimating the support cheaply and tightly, the FSP methods can compete strongly on computation times against stochastic simulation based methods. We have shown experimentally that the number of realisations needed could be three to four orders of magnitude larger than the minimal support size. In such cases, it would be cheaper to look for the minimal domain and solve the CME over it than to compute realisations. For small dimensional problems, the stochastic simulation methods have been out-performed by FSP type methods. We verified that this is because the support is easier to find in smaller dimensional problems.

We observed that by adaptively choosing the domain, we can compute approximations of large CME problems in seconds (OFSP). In numerical experiments, the OFSP method performed four to ten times faster than the conventional FSP method. The same examples have been computed in the literature and have taken thousands of seconds to compute, the significant factor contributing to their lack of speed being hyper-rectangular domain selection. Hence, by adaptively selecting our domain using the OFSP method, we have reduced our computation time to a few hundred seconds.

We have shown that the CME problem can be divided into smaller problems for distributed computing (PFSP). We parallelised the CME by simply partitioning the domain into non-intersecting sets and computing each partition on a
different core. In numerical experiments with a toy problem and simple implementation, we showed that the method of sub-dividing the CME problem used in the PFSP method works and has potential for solving RDME type problems.

Lastly, we proposed the first algorithm (GORDE) which can accurately estimate the domain for a future time step without needing to solve the CME. The GORDE algorithm requires more steps in its computation than the $N$-step method. However, we see in our experiments that the GORDE method selects far fewer points and is four to eight times faster than the $N$-step method. The GORDE algorithm needs an extra step to help remove states as well as add states. With this feature, we conjecture that the GORDE algorithm can provide an optimal order domain for the future time step.

With respect to computation times, we have observed that FSP($N$-step) > FSP(GORDE) > OFSP($N$-step). However, this is also a problem-specific relationship. In this thesis we are only interested in the transient phase of the stochastic process. This is were the approximation is required and here the relationship mentioned above is valid. However, for a system which reaches equilibrium, this ordering might not hold. OFSP($N$-step) would keep truncating and growing the state space, increasing the computation time, while the FSP($N$-step) or GORDE would simply fix the state space and continue time evolutions.

Using the OFSP method to construct an approximation, the GORDE algorithm for finding the domain and the PFSP method to distribute the computation, we proposed a new way of computing an FSP. We proved theoretically that these methods are accurate and also conducted some numerical experiments where the speed improvements compared to literature is shown to be significant. The combination of these three methods can help in reducing the redundancies in existing numerical methods. For example, the domain of the CME problem is chosen to be very large in the current implementation of the wavelet method by Jahnke et. al. This is because the location of the probability is not known a priori. In the wavelet implementation, there is significant wastage due to the presence of points which are close to zero probability. To address this, we can use the GORDE algorithm to estimate the domain for a given time step, following which the wavelet approximation is constructed on that domain. Likewise, in the aggregation framework, the reduced problem is still a CME problem, thus allowing the use of the OFSP, GORDE and PFSP methods to compute the reduced problem faster. This opens up a larger class of biological problems which can be computed by the aggregation method.

The following are the future avenues for research:
• Design a method of compressing the GORDE algorithm which makes its support of optimal order.

• An advanced implementation of the PFSP method which utilises the hardware architecture available in HPC clusters.

• Use CME with respect reaction counts to generate error bounds of empirical distributions generated by stochastic simulation methods.

• Apply the new CME approximations to fit experimental data.

We have proposed new methods which are analytically and numerically sound. This new framework can help speed up the computation of the CME to aid in in-silico simulation and parameter estimation of complex biological systems.
Appendix A

Marginals and Algorithm.

The following is the algorithm of the OFSP implementation using CMEPy [1].

Table A.1: OFSP implementation input parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{\text{final}}$</td>
<td>final time point.</td>
</tr>
<tr>
<td>$t_{\text{start}}$</td>
<td>starting time point.</td>
</tr>
<tr>
<td>$p_0$</td>
<td>initial distribution at $t_{\text{start}}$.</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>global error of approximation at $t_{\text{final}}$.</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>time step.</td>
</tr>
<tr>
<td>$h$</td>
<td>number of time steps before compression.</td>
</tr>
</tbody>
</table>

Figure A.1: Conditional probability of the Goutsias’ model (§3.3.1) evaluated at $t_{\text{final}} = 300$ using OFSP method with error $10^{-4}$. 
Algorithm 11: OFSP implementation using CMEPy \[1\]

\[\text{Algorithm 11: OFSP implementation using CMEPy}\]

\textbf{input}: \(t_{\text{final}}, t_{\text{start}}, \varepsilon, \Delta t, p_0, h\)

\textbf{output}: \(p_{t_{\text{final}}}\)

\begin{algorithmic}
\STATE \textbf{begin}
\STATE \hspace{1cm} steps \(\leftarrow\) \(\frac{t_{\text{final}}-t_{\text{start}}}{\Delta t}\)
\STATE \hspace{1cm} \(\varepsilon_{\text{step}} \leftarrow \frac{\varepsilon}{\text{steps}}\)
\STATE \hspace{1cm} \(t \leftarrow t_{\text{start}}\)
\STATE \hspace{1cm} \(p \leftarrow p_0\)
\STATE \hspace{1cm} \(\text{step}_{counter} \leftarrow 0\)
\FOR {\(t \leq t_{\text{final}}\)}
\STATE \hspace{1cm} \(p_t \leftarrow \text{FSP}(p, \varepsilon_{\text{step}}/2, \Delta t)\)
\IF {\(\text{step}_{counter} == h\)}
\STATE \hspace{1cm} \(p_t \leftarrow \text{compress}(p_t, h\varepsilon_{\text{step}}/2)\) \hspace{1cm} (c.f. Algorithm 5)
\STATE \hspace{1cm} \(\text{step}_{counter} = 0\)
\ENDIF
\STATE \hspace{1cm} \(t \leftarrow t + \Delta t\)
\STATE \hspace{1cm} \(\text{step}_{counter} = \text{step}_{counter} + 1\)
\ENDFOR
\STATE \hspace{1cm} \(\text{return} p_{t_{\text{final}}}\)
\STATE \textbf{end}
\end{algorithmic}

Figure A.2: Conditional probability of the Goutsias’ model (§3.3.1) evaluated at \(t_{final} = 300\) using SSA with \(10^6\) realisations.
Figure A.3: The difference the OFSP approximation and the SSA approximation of the Goutsias’s model (§3.3.1) at $t_{final} = 300$.

Figure A.4: Conditional probability of the G-K Switch (§3.3.2) evaluated at $t_{final} = 20$ using OFSP method with error $10^{-4}$. 
APPENDIX A. MARGINALS AND ALGORITHM.

Figure A.5: Conditional probability of the G-K Switch (§3.3.2) evaluated at $t_{\text{final}} = 20$ using SSA with $10^6$ realisations.

Figure A.6: The difference the OFSP approximation and the SSA approximation of the G-K Switch (§3.3.2) at $t_{\text{final}} = 20$. 
Appendix B

Transition Probability

We show how the transition probability is derived for a Poisson process. The derivations here are not new but can be found in popular stochastic processes literature [15].

Let $X(t)$ be a stochastic process and $\lambda$ its rate coefficient. Then for $t, \Delta t > 0$ we define

$$N(t, \Delta t) = X(t + \Delta t) - X(t),$$

as number of events which have occurred between $t$ and $t + \Delta t$.

By the definition of a Poisson process we have that,

$$P[N(t, \Delta t) = k] = \frac{(\lambda \Delta t)^k e^{-\lambda \Delta t}}{k!}.$$

We can now derive the probability of events occurring. The probability of one event occurring in time $\Delta t$ is

$$P[N(t, \Delta t) = 1] = \lambda \Delta t e^{-\lambda \Delta t} = \lambda \Delta t + O(\Delta t).$$

Similarly the probability of no event occurring is,

$$P[N(t, \Delta t) = 0] = e^{-\lambda \Delta t} = 1 - \lambda \Delta t + O(\Delta t).$$

We can now derive the probability of more than one event occurring in $\Delta t$, it is simply one minus the sum of the no events and one event occurring. This gives us

$$P[N(t, \Delta t) > 1] = O(\Delta t).$$

For the multiple processes case, let $X(t)$ be the combination of $n$ independent Poisson processes $X_1, \ldots, X_n$, with rate $\lambda_1, \ldots, \lambda_n$. Let

$$N(t, \Delta t) = X(t + \Delta t) - X(t),$$
denote the number of events which have occurred in the time interval \([t, t + \Delta t]\). Since the processes are independent, we define for \(i = 1, \ldots, n\),

\[ N_i(t, \Delta t) = X_i(t + \Delta t) - X_i(t), \]

as the number of events which have occurred in the time interval \([t, t + \Delta t]\) in the \(i\)th process.

For each individual process \(X_i\), with \(i = 1, \ldots, n\), we have,

\[ P[N_i(t, \Delta t) = k] = \frac{(\lambda_i \Delta t)^k e^{-\lambda_i \Delta t}}{k!}. \]

Then for an interval of \([t, t + \Delta t]\) we can derive the probability of no event, one event and more than one event occurring.

\[ P[N(t, \Delta t) = 0] = \prod_{i=1}^{n} P[N_i(t, \Delta t) = 0], \]

\[ = \prod_{i=1}^{n} (1 - \lambda_i \Delta t + O(\Delta t)), \]

\[ = 1 - \sum_{i=1}^{n} \lambda_i + O(\Delta t). \]

\[ P[N(t, \Delta t) = 1] = \sum_{i=1}^{n} P[N_i(t, t + \Delta t) = 1] \times P[N_j(t, \Delta t) = 0; j \neq i], \]

\[ = \sum_{i=1}^{n} (\lambda_i \Delta t + O(\Delta t)) \prod_{i \neq j} (1 - \lambda_j \Delta t + O(\Delta t)), \]

\[ = \sum_{i=1}^{n} \lambda_i \Delta t + O(\Delta t). \]

Similar to the one Poisson process case, if we subtract the sum of probabilities of no reaction and one reaction, we will get the probability of two or more reactions occurring in the interval \([t, t + \Delta t]\).

\[ P[N(t, \Delta t) > 1] = O(\Delta t). \]

In section 2.1 we discuss the Kurtz formulation (2.12), a stochastic formulation which is made up random time change Poisson processes \(Y\), where

\[ X(t) = Y \left( \int_0^t \alpha(X(s))ds \right), \]
with being a $\alpha$ is a propensity function.

The transition probabilities for the random time change Poisson processes are approximated by homogeneous Poisson processes for small $\Delta t$ \[35\]. That is,

\[
P[N(t, \Delta t) = 1] \approx \alpha(X(t))\Delta t, \\
P[N(t, \Delta t) = 0] \approx 1 - \alpha(X(t))\Delta t + O(\Delta t), \\
P[N(t, \Delta t) > 1] \approx O(\Delta t).
\]

If the stochastic process is given as,

\[
X(t) = \sum_{i=1}^{N_r} Y_i \left( \int_0^t \alpha_i(X(s))ds \right) v_i,
\]

a combination of random time change Poisson processes, with $v_i$ an $N_s \in \mathbb{N}$ dimensional stoichiometric vector, then for small $\Delta t$, we approximate the transition probabilities by,

\[
P[\|X(t + \Delta t) - X(t)\|_1 = 1] \approx \sum_{i=1}^{N_r} \alpha_i(X(t))\Delta t, \\
P[\|X(t + \Delta t) - X(t)\|_1 = 0] \approx 1 - \sum_{i=1}^{N_r} \alpha_i(X(t))\Delta t + O(\Delta t), \\
P[\|X(t + \Delta t) - X(t)\|_1 > 1] \approx O(\Delta t).
\]

Our focus is to substitute these transitions into the Chapman-Kolmogorov equations and let $\Delta t$ tend to zero. Hence these approximations are sufficient for describing the transition probabilities of the Kurtz formulation \[2.12\].
Bibliography


