Numerical Methods for
the Chemical Master Equation

Jingwei Zhang

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY
in
Mathematics

APPROVED:

Layne T. Watson, Chair
Christopher A. Beattie
Terry L. Herdman
Tao Lin
Calvin J. Ribbens

December 2, 2009
Blacksburg, Virginia

Key words: Chemical Master Equation, Stochastic Simulation Algorithm, Uniformization/Randomization Method, Parallel Computing, Aggregation/Disaggregation, Collocation Method, Radial Basis Function, Shepard Algorithm, M-estimation

Numerical Methods for
the Chemical Master Equation

Jingwei Zhang

(ABSTRACT)

The chemical master equation, formulated on the Markov assumption of underlying chemical kinetics, offers an accurate stochastic description of general chemical reaction systems on the mesoscopic scale. The chemical master equation is especially useful when formulating mathematical models of gene regulatory networks and protein-protein interaction networks, where the numbers of molecules of most species are around tens or hundreds. However, solving the master equation directly suffers from the so called "curse of dimensionality" issue. This thesis first tries to study the numerical properties of the master equation using existing numerical methods and parallel machines. Next, approximation algorithms, namely the adaptive aggregation method and the radial basis function collocation method, are proposed as new paths to resolve the "curse of dimensionality". Several numerical results are presented to illustrate the promises and potential problems of these new algorithms. Comparisons with other numerical methods like Monte Carlo methods are also included. Development and analysis of the linear Shepard algorithm and its variants, all of which could be used for high dimensional scattered data interpolation problems, are also included here, as a candidate to help solve the master equation by building surrogate models in high dimensions.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Layne T. Watson, for serving as my research advisor and committee chairperson, for demonstrating how to do research, for teaching me how to write research papers and correcting thoroughly every English grammar mistake I have made, and for giving me opportunities to attend various conferences. I would like to thank Dr. Cao Yang for working with me on many research problems and writing recommendation letters. I would like to thank Drs. Masha Sosonkina, William Thacker, and Jeffery Birch for their advice on my research topics. I would also like to thank the JigCell project group for giving me chances to present and discuss my research on bioinformatics.

I would like to thank Dr. Lin Tao for guiding me through my first year of graduate study at Virginia Tech, serving on my thesis committee, and inviting me home for dinner so many times. I would like to thank Dr. Ekkehard Sachs for giving me great help and advice on mathematical research and best wishes when I decided to switch research topic and advisor. I would like to thank Dr. Calvin Ribbens for the opportunity to teach the undergraduate numerical methods course and serving on my thesis committee. I would like to thank Dr. Christopher Beattie for his numerical linear algebra expertise and guidance on my research work and thank him for serving on my committee, for that I would also like to thank Dr. Terry Herdman. I would like to thank Dr. Traian Iliescu for providing me advice on graduate research and recommendation letters.

I would like to thank Dr. William Greenberg and Ms. Terri Bourdon for their sincere help on teaching Math 1224 recitation sessions and all my teaching assistant colleagues I have worked with in the Math Emporium and recitation sessions for their advice and help.

I would like to take this chance to thank my former and current labmates, Rhonda, Manjula, Sean, Ted, Pengyuan, Zhenwei, Shubhangi, David, and Nicholas for being great labmates and friends, for inviting me to their homes and sharing food. I would also like to thank my many roommates and all my friends in Blacksburg, for making my life in this small American town not so miserable. I would especially like to thank my girl friend, Kejia Wu, for making my last semester at Virginia Tech the happiest time in my life.

Finally, I would to like to acknowledge my family, especially my parents for their support on every little step of this twenty-two year long trek of education, from kindergarten through graduate school, from China to the USA.

TABLE OF CONTENTS.

LIST OF FIGURES.

LIST OF TABLES.

Chapter 1: INTRODUCTION

There are many mathematical ways to describe the dynamics of chemical reactions. One of them is the deterministic *reaction rate equation* (RRE), which is accurate only when the number of reacting molecules is large enough to allow a continuum point of view. In the modeling of gene regulatory networks and protein-protein interaction networks, the number of molecules of a given chemical species is typically on the order of hundreds. In such a situation, the randomness in the system usually cannot be ignored. Thus, one is forced to adopt a stochastic description.

The chemical master equation (CME) ([95], [33]) is such a stochastic description, a mathematical formulation for the time evolution of the probability of the chemical system to occupy every possible discrete state, derived from the Markov property of the underlying chemical kinetics. If the chemical system is determined by specifying the number of molecules of each species, then the master equation governs the dynamics of the probability distribution for the system. However, it is well known that such a description suffers from the "curse of dimensionality", i.e., each new species adds one dimension to the problem, thus, the computational complexity grows exponentially.

Monte Carlo algorithms are used to analyze the CME without suffering from the curse. These methods simulate one trajectory at a time. Even though simulating one trajectory might be performed relatively cheaply, many trajectories need to be simulated in order to estimate statistical parameters and probability distributions accurately.

Recently, there has been considerable work on solving the master equation directly. This dissertation aims to contribute some new ideas on numerical methods for the CME.

The rest of this chapter is arranged as follows. Section 1.1 talks about deterministic chemical kinetics. Section 1.2 describes the basics of stochastic chemical kinetics, including the derivation of the chemical master equation. Section 1.3 focuses on numerical methods for stochastic chemical kinetics, including the stochastic simulation algorithms and other methods. Section 1.4 concludes.

1.1 Deterministic Chemical Kinetics

Assume a fixed volume V contains a well-stirred mixture of D chemical species $S_1, S_2, \ldots, S_D$, reacting through M chemical reaction channels $R_1, R_2, \ldots, R_M$. Well-stirred here has two implications: the system of molecules is *homogeneous*, meaning the probability of finding any randomly selected molecule inside any volume ΔV is $\Delta V/V$ and the system of molecules is in *thermal equilibrium*, meaning the macroscopic thermal observables do not change over time.

The traditional way to describe the "deterministic" time evolution of the molecular population levels given the initial condition is the ordinary differential equations (ODEs) called the reaction rate equations.

The reaction rate equation is derived under the law of mass action kinetics: the speed at which each reaction proceeds is proportional to the concentrations of its reactants.

For example, consider the Lotka reactions,

$$\begin{array}{lccc} R_1) & S_1 & \xrightarrow{k_1} & 2S_1, \\ R_2) & S_1 + S_2 & \xrightarrow{k_2} & 2S_2, \\ R_3) & S_2 & \xrightarrow{k_3} & Z, \end{array}$$

with $k_1 = 10$, $k_2 = 0.01$, and $k_3 = 10$. Let the "continuous" variables $Y_i = [S_i]$, $i = 1, 2$, denote the concentrations of species S_1 and S_2. The law of mass action kinetics states the reaction rates are proportional to the number of molecule collisions, which is proportional to the product of reactant concentrations:

$$\text{reaction rates for } \begin{pmatrix} R_1 \\ R_2 \\ R_3 \end{pmatrix} \text{ are } \begin{pmatrix} k_1 Y_1 \\ k_2 Y_1 Y_2 \\ k_3 Y_2 \end{pmatrix}.$$

Let ν be the stoichiometric matrix of the Lotka reactions, where ν_{ij} is the change of the stoichiometric coefficient of species i due to reaction j. Then,

$$\nu = \begin{array}{c} \\ S_1 \\ S_2 \end{array} \begin{array}{c} \begin{array}{ccc} R_1 & R_2 & R_3 \end{array} \\ \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix} \end{array}.$$

The right hand side of the deterministic ODE can then be expressed as the product of the stoichiometric matrix times the reaction rates vector,

$$\frac{d}{dt}Y(t) = \nu \cdot R\big(Y(t)\big),$$

i.e.,

$$\frac{d}{dt}\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} k_1 Y_1 \\ k_2 Y_1 Y_2 \\ k_3 Y_2 \end{pmatrix}.$$

1.2 Stochastic Chemical Kinetics

In many cases the dynamics of the chemical system can be treated as a continuous, deterministic process to an acceptable degree of accuracy. However, in certain other types of studies, the inability to describe the fluctuations in the molecular population levels of the reaction rate equation can be very serious. For example, for systems with very small numbers of molecules, the underlying molecular dynamics fluctuations are not smoothed out by statistical averaging, and the time evolutions of such systems are therefore stochastic.

Under the same problem setting and assumptions, the system is now described in terms of the time evolution of the "discrete" state vector $x(t) = \big(x_1(t), x_2(t), \ldots, x_D(t)\big)$, where $x_i(t)$ is the number of molecules of species S_i at time t. Then, the concentration of species S_i is $Y_i(t) = \frac{\langle x_i(t)\rangle}{V}$, where $\langle\cdot\rangle$ denotes an average over an ensemble of identical systems.

The mathematical way to describe the "stochastic" time evolution of the state vector $x(t)$ given the initial condition is the chemical master equation.

Using the derivation in [32], for each reaction channel R_j there exists a constant c_j such that the average probability at which a particular combination of R_j reactants will react in the infinitesimal time interval $[t, t+dt)$ is $c_j dt$. Hence, given the time t and the state vector $x(t)$, the probability $a_j(x(t))dt$ that the reaction R_j happens in $[t, t+dt)$ is $c_j dt$ times the total number of reactant combinations. $a_j(x(t))$ is called the propensity function.

For example, the Lotka reactions and their propensities are

$$\begin{aligned}
R_1) &\quad S_1 \xrightarrow{k_1} 2S_1, &\quad a_1 &= c_1 x_1, \\
R_2) &\quad S_1 + S_2 \xrightarrow{k_2} 2S_2, &\quad a_2 &= c_2 x_1 x_2, \\
R_3) &\quad S_2 \xrightarrow{k_3} Z, &\quad a_3 &= c_3 x_2.
\end{aligned}$$

In cases where multiple reactants are of the same type, like $S_1 + S_1 \longrightarrow \cdots$, $a_j = c_j \frac{x_1(x_1-1)}{2}$, since $\frac{x_1(x_1-1)}{2}$ is the number of distinctive pairs of molecules of species S_1.

1.2.1 Connection with Deterministic Kinetics

For large numbers of molecules, the Poisson mean number of the reaction R_j likely to happen in $[t, t+dt)$ is $a_j(x(t))dt$ [32]. The ensemble average $\langle a_j(x(t))dt\rangle \approx a_j(\langle x(t)\rangle)dt$ for large molecular populations (where $x(t)$ is nearly deterministic). On the other hand, the deterministic reaction rate expresses the average number of reactions that happens per unit volume per unit time. Therefore, for the reaction R_j, the propensity function a_j is related to the reaction rate constant k_j.

For example, in the Lotka reactions:

$$R_1) \quad S_1 \quad \xrightarrow{k_1} \quad 2S_1, \quad a_1 = c_1 x_1.$$

Therefore,

$$k_1 Y_1 = \frac{a_1(\langle (x_1, x_2)\rangle)}{V} = \frac{c_1\langle x_1\rangle}{V} = c_1 Y_1$$
$$\Longrightarrow k_1 = c_1.$$

$$R_2) \quad S_1 + S_2 \quad \xrightarrow{k_2} \quad 2S_2, \quad a_2 = c_2 x_1 x_2.$$

Therefore,

$$k_2 Y_1 Y_2 = \frac{a_2(\langle (x_1, x_2)\rangle)}{V} = \frac{c_2\langle x_1\rangle\langle x_2\rangle}{V} = c_2 V Y_1 Y_2$$
$$\Longrightarrow k_2 = c_2 V.$$

$$R_3) \quad S_2 \quad \xrightarrow{k_3} \quad Z, \quad a_3 = c_3 x_2.$$

Therefore,

$$k_3 Y_2 = \frac{a_3(\langle (x_1, x_2)\rangle)}{V} = \frac{c_3\langle x_2\rangle}{V} = c_3 Y_2$$
$$\Longrightarrow k_3 = c_3.$$

Also,

$$S_1 + S_1 \quad \xrightarrow{k_j} \quad \cdots, \quad a_j = c_j \tfrac{x_1(x_1-1)}{2}.$$

Therefore,

$$k_j Y_1^2 = \frac{a_j(\langle (x_1, x_2)\rangle)}{V} = \frac{c_j}{V}\frac{\langle x_1\rangle(\langle x_1\rangle - 1)}{2} \approx \frac{c_j V}{2}\left(\frac{\langle x_1\rangle}{V}\right)^2 = \frac{c_j V}{2} Y_1^2$$
$$\Longrightarrow k_j = \frac{c_j V}{2}.$$

1.2.2 Grand Probability Function and the CME

The grand (transition) probability function $p(x, t|x^0, t_0)$ denotes the probability that there will be $x = (x_1, x_2, \ldots, x_D)$ molecules of each species at time t in V, given that the numbers of molecules is x^0 at t_0. The initial condition x^0, t_0 is often suppressed to simplify the notation $(p(x, t))$.

The (chemical) master equation is the time-evolution equation for the grand probability function, using the Markov property, which states that the conditional probability for the event (x^n, t_n) given the full history of the system satisfies

$$p(x^n, t_n|x^{n-1}, t_{n-1}; \ldots; x^0, t_0) = p(x^n, t_n|x^{n-1}, t_{n-1}),$$

meaning the dependence of the present (x^n, t_n) on past events can be captured solely by the dependence on the previous state (x^{n-1}, t_{n-1}).

Although the Markov property is not exactly fulfilled for any given physical/chemical system, it can often serve as an accurate approximation. One important consequence of the Markov assumption is the *Chapman-Kolmogorov* equation,

$$p(x^2, t_2|x^0, t_0) = \sum_{x_1} p(x^2, t_2|x^1, t_1)p(x^1, t_1|x^0, t_0).$$

The master equation can be derived directly from the Chapman-Kolmogorov equation. The time derivative of the grand probability function is defined as

$$\frac{\partial}{\partial t}p(x, t|x^0, t_0) = \lim_{\Delta t \to 0} \frac{p(x, t + \Delta t) - p(x, t)}{\Delta t}.$$

Introduce a dummy variable y using the Chapman-Kolmogorov equation,

$$\frac{\partial}{\partial t}p(x, t|x^0, t_0) = \lim_{\Delta t \to 0} \frac{\sum_y p(x, t + \Delta t|y, t)p(y, t) - p(y, t + \Delta t|x, t)p(x, t)}{\Delta t}.$$

Let $W(x^2|x^1) = \lim_{\Delta t \to 0} p(x^2, t + \Delta t|x^1, t)/\Delta t$ denote the *transition probability per unit time* from state x^1 to x^2. The above formula for the time derivative can be simplified as

$$\frac{\partial}{\partial t}p(x, t|x^0, t_0) = \sum_y W(x|y)p(y, t) - W(y|x)p(x, t).$$

For chemical systems, $W(x^2|x^1)$ is nonzero if and only if there is chemical reaction connecting x^2 with x^1. The reaction $R_j)\quad x-\nu_j \longrightarrow x$ (ν_j is the j-th column of matrix ν) happens with probability $a_j(x-\nu_j)dt$ in interval $[t, t+dt)$, which implies that $W(x|x-\nu_j) = a_j(x-\nu_j)$. Therefore

$$\frac{\partial}{\partial t}p(x,t|x^0,t_0) = \sum_{j=1}^{M} a_j(x-\nu_j)p(x-\nu_j,t) - a_j(x)p(x,t). \qquad \text{(CME)}$$

Another way to derive the master equation is to write $p(x, t+dt)$ as the sum of the probabilities of the $1+M$ different ways in which the system can reach the state x at time $t+dt$:

$$p(x,t+dt) = p(x,t) \times \left(1 - \sum_{j=1}^{M} a_j(x)dt\right) + \sum_{j=1}^{M} p(x-\nu_j,t) \times a_j(x-\nu_j)dt,$$

where the first term represents the probability that no reaction occurs during $[t, t+dt)$ and the system remains in state x, while each term in the second summation is the probability that one reaction R_j occurs in $[t, t+dt)$ and changes the state $x-\nu_j \longrightarrow x$. Rearranging this formula can also verify the master equation.

Taking the sum of all possible states on both sides of the master equation proves that the master equation conserves probability. Next, using the master equation to compute the mean value $\langle x(t)\rangle = \sum xp(x,t)$ yields

$$\frac{d}{dt}\langle x(t)\rangle = \sum_{j=1}^{M} \nu_j \langle a_j(x(t))\rangle = \sum_{j=1}^{M} \nu_j a_j(\langle x(t)\rangle),$$

which is equivalent to the reaction rate equation if all the propensity functions $a_j(\cdot)$ are linear (the last equality requires the linearity).

1.3 Numerical Methods for the CME

1.3.1 Direct Methods for the CME

When the state space $\{x^1, x^2, \ldots\}$ (each element x^i is a distinct D-dimensional molecular population vector) is chosen, the chemical master equation can be rewritten into an infinite linear system of ODEs (ordinary differential equation),

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

where $P(t)$ is the complete probability row vector at time t, $P(t) = (p(x^1, t), p(x^2, t), \ldots)$. The time-independent matrix A is defined from the nonnegative propensity functions,

$$A_{ij} = \begin{cases} -\sum_{\mu=1}^{M} a_\mu(x^i), & \text{for } i = j, \\ a_\mu(x^i), & \text{for } j \text{ such that } x^j = x^i + \nu_\mu, \\ 0, & \text{otherwise.} \end{cases}$$

It can be seen from the definition above that A is extremely sparse with at most $M + 1$ nonzero elements in each row and each row of A sums up to zero.

In theory, the size of the matrix A is infinite, but in any physical system, the number of molecules of each species is bounded. To compute the solution $P(t)$ numerically, the CME is often truncated to a finite state problem, restricting the state space of the CME to a finite domain, which is also large enough to represent the true physical solution. However, for most chemical systems, the size of the CME after truncation is often huge, on the order of 10^5 to 10^9, which makes solving the chemical master equation numerically intensive. The CME illustrates "the curse of dimensionality", which states that the computational complexity grows exponentially with the dimension (the number of reacting species here), unless other assumptions are made.

Numerical approaches solving the CME directly as a linear differential equation can be roughly categorized into two groups, grid-based methods and Galerkin-based ones [47]. The grid-based methods, which explicitly truncate/aggregate the probability distribution into some finite, smaller domains, include the finite state projection (FSP) method [61], adaptive

FSP with Krylov-based exponential computation [54], (adaptive) sparse grids methods [40, 41], and others [25, 102].

Grid-based methods first grid the whole domain space into pairwise disjoint subsets, where each subset may contain just one state or tens to hundreds of states. Then, based on some given criteria, a number of these subsets are selected and their union forms the computation domain. For example, Zhang et al. [102] use a few runs of SSA simulations to select potential subsets, precisely, the union of those subsets that have been touched by at least one simulation run is used as the computation domain. All states in the same selected subset are then aggregated into one single state by the aggregation operator, thus reducing the problem size. The solution $P(t)$ can be recovered from the solution to the reduced problem by applying the disaggregation operator. In (adaptive) sparse grids methods, a linear combination of several aggregation/disaggregation operator pairs is applied to achieve maximum reduction of the problem size. However, in most of these grid-based approaches, disaggregation operators are defined by piecewise constant/linear polynomial functions.

The Galerkin-based method, where the probability distribution is represented in some (adaptive) prechosen finite dimensional projection space, was first proposed by Engblom [23], and coincidentally was mentioned as early as two decades ago by Deuflhard and Wulkow [19]. They both choose to project the discrete solution onto some subspace formed by a time-dependent orthogonal basis of discrete Charlier polynomials. The thusly constructed "global" Galerkin methods are more suitable for unimodal solutions, since the Charlier polynomials are used globally. For bimodal solutions or solutions with more than two peaks, localized polynomials/basis functions seem more appropriate [20]. Jahnke and Huisinga [47] tried a different Dirac-Frenkel-McLachlan variational principle on the CME. Most of these Galerkin-based approaches are based on restrictive orthogonal basis functions.

Another numerically oriented approach is to approximate the operator A in the master equation by its second order Taylor expansion, which gives the *Fokker-Planck equation*,

$$\frac{\partial}{\partial t}p(x,t) = -\sum_{j=1}^{M} \nu_j^T \nabla_x (a_j(x)p(x,t)) + \frac{1}{2} v_j^T \nabla_x \left(v_j^T \nabla_x (a_j(x)p(x,t)) \right).$$

The Fokker-Planck equation is a D-dimensional parabolic partial differential equation (PDE), that can be solved by well established numerical PDE methods. Computational effort is saved since the spatial discretization for numerical PDE methods can be much coarser than the actual state space. However, it is often difficult to determine a priori how well the continuous Fokker-Planck equation approximates the discrete master equation.

1.3.2 The Direct Stochastic Simulation Algorithm

The stochastic simulation algorithm (SSA) [32] offers an alternative way to approximate the grand probability function besides solving the CME directly. Define the function $p(\tau, \mu)$ such that $p(\tau, \mu)d\tau$ is the probability that, given the state x at time t, the next reaction in the system will occur in the infinitesimal time interval $[t + \tau, t + \tau + d\tau)$, and will be R_μ. The probability function $p(\tau, \mu)$ can be decomposed as the product of the probability function $p_0(\tau)$, the probability that, given the state x at time t, no reaction will occur in the time interval $[t, t + \tau)$, times $a_\mu d\tau$, the probability that the reaction R_μ will occur in the time interval $[t + \tau, t + \tau + d\tau)$, i.e.,

$$p(\tau, \mu)d\tau = p_0(\tau)a_\mu d\tau.$$

Note that the probability that no reaction will occur in the infinitesimal time interval $[t, t + dt)$ is $p_0(dt) = (1 - \sum_j a_j dt)$. Let $a_0 = \sum_j a_j$, then

$$\left.\begin{aligned} p_0(\tau + d\tau) &= p_0(\tau)(1 - a_0 d\tau) \\ p_0(0) &= 1 \end{aligned}\right\} \Longrightarrow p_0(\tau) = e^{-a_0\tau}$$

and

$$p(\tau, \mu) = e^{-a_0\tau}a_\mu = \underbrace{a_0 e^{-a_0\tau}}_{p_t(\tau)} \cdot \underbrace{(a_\mu/a_0)}_{p_r(\mu)},$$

where $p_t(\tau)$ and $p_r(\mu)$ are the probability density/mass functions for the random variables τ and μ, respectively.

The direct stochastic simulation algorithm is stated as follows.

Step 1. Set the time variable $t := 0$ and the state variable x to the initial state.

Step 2. Calculate the propensity functions $a_j(x)$, $1 \le j \le M$, for the current state x and the sum $a_0(x) = \sum_j a_j(x)$.

Step 3. Generate random numbers τ and μ from the distributions with probability density/mass functions $p_t(\tau)$ and $p_r(\mu)$. One way to achieve this is to first generate two uniform $U(0, 1)$ random numbers r_1 and r_2 and then choose the next reaction time by

$$\tau = (1/a_0)\ln(1/r_1)$$

and the reaction channel μ as the integer that satisfies the inequality

$$\sum_{j=1}^{\mu-1} a_j < r_2 a_0 \leq \sum_{j=1}^{\mu} a_j.$$

Step 4. Update the system, $t := t + \tau$ and $x := x + \nu_\mu$. Repeat from Step 2 until the final time t_f is reached.

The SSA algorithm stated here is exact, in the sense that the algorithm generates a sample trajectory consistent with the CME, thus, the grand probability function $p(x, t|x^0, t_0)$ can be estimated from a set of trajectories starting from the same initial condition (x^0, t_0). However, Monte Carlo methods like SSA converge very slowly, meaning many trajectories are needed to estimate statistical parameters and probability distributions accurately. Moreover, since the SSA is an explicit method, simulating one trajectory itself may not be easy in some cases.

1.3.3 Tau-leaping Method and Other Monte Carlo Methods

In order to speed up the original SSA algorithm, improvements have been made by adopting different approximation techniques. One of the most famous and promising approaches is the tau-leaping method [35], which uses the Poisson approximation to "leap over" many reactions.

Assume that the time step τ is short enough such that the expected change in molecular numbers in the time interval $[t, t+\tau)$ leads to a negligible expected change of the propensity functions, i.e.,

$$a_j(x(t+\tau)) \approx a_j(x(t)), \quad 1 \leq j \leq M.$$

Then the number of firings of reaction R_j in $[t, t+\tau)$ is a Poisson random variable with mean $a_j(x(t))\tau$ and is independent from all other reactions.

The algorithm for the explicit tau-leaping method is similar to the SSA, except that at Step 3, τ is chosen deterministically to satisfy the "leap condition" and for each reaction channel j, the number of firings k_j is generated as a Poisson random number $\mathcal{P}\big(a_j(x(t))\tau\big)$ with mean $a_j(x(t))\tau$. At Step 4 the system is updated with $t = t + \tau$ and $x = x + \sum_j k_j \nu_j$.

Another way to express the explicit tau-leaping method is

$$x(t+\tau) = x(t) + \sum_{j=1}^{M} \mathcal{P}\big(a_j(x(t))\tau\big)\nu_j.$$

At a coarser scale, suppose that the leap interval τ spans a very large number of firings of each reaction, yet only insignificant changes in each propensity function are induced, which is reasonable for large numbers of molecules. Then, the Poisson random variable $\mathcal{P}\big(a_j\tau\big)$ is well approximated by the normal random variable $\mathcal{N}\big(a_j\tau, a_j\tau\big)$, implying the *Langevin leaping formula*,

$$\begin{aligned} x(t+\tau) &= x(t) + \sum_{j=1}^{M} \mathcal{N}\big(a_j(x(t))\tau, a_j(x(t))\tau\big)\nu_j \\ &= x(t) + \sum_{j=1}^{M} a_j(x(t))\tau\nu_j + \sum_{j=1}^{M} \sqrt{a_j(x(t))\tau}\mathcal{N}_j(0,1)\nu_j. \end{aligned}$$

Note that in this formula $x(t)$ is a real function. The Langevin leaping formula is actually equivalent to the *chemical Langevin equation (CLE)*,

$$dx(t) = \sum_{j=1}^{M} a_j(x(t))\nu_j dt + \sum_{j=1}^{M} \sqrt{a_j(x(t))}\nu_j dW_j,$$

where W_j is a Wiener process and $dW_j/dt = \mathcal{N}_j(0, 1/dt)$ is Gaussian white noise. Note that for very large numbers of molecules, the stochastic term $\sqrt{a_j(x(t))}\nu_j$ is much smaller than the deterministic term $a_j(x(t))\nu_j$. In the limit the stochastic fluctuations in the CLE become negligible and the CLE approximates the RRE

$$\frac{dx(t)}{dt} = \sum_{j=1}^{M} a_j(x(t))\nu_j.$$

The original explicit tau-leaping method is not very efficient when stiffness is present (where some reactions occur much faster than the others, and the fast reactions force τ to be very small). Implicit tau-leaping methods have been proposed to solve this issue [71],

$$x(t+\tau) = x(t) + \sum_{j=1}^{M} a_j(x(t+\tau))\tau\nu_j + \sum_{j=1}^{M} \left(\mathcal{P}(a_j(x(t))\tau) - a_j(x(t))\tau\right)\nu_j.$$

Another way to deal with the stiffness efficiently and to accelerate the original SSA algorithm is to make use of stochastic versions of the quasi steady state or partial equilibrium assumptions. In the deterministic case, the quasi steady state approximation assumes that at some time scale instantaneous rates of change for some intermediate species are approximately zero, while the partial equilibrium approximation assumes that some fast reaction channels are always in equilibrium. The former one was extended to the stochastic quasi steady state approximation (SQSSA) [70] and the latter one was used to develop the slow-scale SSA method [10].

To illustrate the quasi steady state approximation (QSSA) and the partial equilibrium assumption, consider the system of reactions

$$\begin{aligned} \frac{d[A]}{dt} &= \epsilon^{-1} f([A],[B],[C],\ldots) && \text{fast,} \\ \frac{d[B]}{dt} &= g([A],[B],[C],\ldots) && \text{intermediate,} \\ \frac{d[C]}{dt} &= \epsilon h([A],[B],[C],\ldots) && \text{slow,} \end{aligned}$$

where $0 < \epsilon \ll 1$. For the slow reactant C, assume $d[C]/dt \approx 0$, $[C] \approx$ constant (quasi steady state). For the fast reactant A, assume that $[A]$ changes very rapidly about the mean $\langle [A] \rangle$, which is nearly constant in time (partial equilibrium). Then the system could be simplified to involve only the intermediate reaction for $[B]$, taking $[C]$ constant and $[A] = \langle [A] \rangle$.

1.4 Conclusions

Given a certain error bound ϵ, it is well known that the computational work for the Monte Carlo simulation methods like SSA is on the order of ϵ^{-2}. Recall that the CME is essentially a D-dimensional deterministic discrete partial differential equation (PDE), where D is the number of species. For traditional PDE solvers with k-th order of accuracy $\epsilon = cN^{-k/D}$, where c is a constant and N is the number of unknowns for the numerical method in D dimensions. Assume that the computational effort is a linear function of N, then the computational effort is on the order of $\epsilon^{-D/k}$.

Fairly speaking, solving the CME directly in full as discrete PDE is not the best way in high dimensions and/or when only rough estimates of some statistics are needed, compared to Monte Carlo methods. Nevertheless, in the following chapters, several methods are proposed to push the limits of direct solvers.

Chapter 2: A MODIFIED UNIFORMIZATION METHOD FOR THE CME

2.1 Introduction

Basically, numerical approaches on solving the master equation fit in two categories: an ordinary differential equation (ODE) point of view or a partial differential equation (PDE) point of view. From an ODE point of view, one simply integrates the master equation in time. The problem is that the ODE system dimension is often huge, so in order to make this approach feasible, many state approximation techniques have been proposed, like the sparse grids technique [40] and the finite state projection algorithm [61].

From a PDE point of view, the master equation is just a special kind of parabolic partial differential equation. The difficulty lies in that the state space is discrete. Another alternative is solving the Fokker-Planck partial differential equation ([89], [24]); the Fokker-Planck equation can be regarded as a continuous approximation of the master equation. However, it is often difficult to determine a priori how good the approximation would be.

Taking the ODE point of view, success in solving the master equation depends on evaluating the matrix exponential series. In principle, the exponential of a matrix can be computed in many ways, but in practice, taking computational stability and efficiency into account, none of them are completely satisfactory [59]. Hence, the most appropriate method should be based upon particular properties of the matrix.

Uniformization or Jensen's Method [48, 90] is a special technique devised to compute the exponential of the infinitesimal generator of almost any continuous time Markov chain. It has a natural stochastic/probabilistic interpretation. It is easy to implement, only involves matrix-vector multiplication and is numerically stable. There is also a simple error bound for the matrix exponential approximation [39]. Conclusively, for most non-stiff problems, uniformization provides an accurate and economical numerical solution. Unfortunately, for stiff problems, often the case in biological systems, it is computationally inefficient.

To deal with stiff problems, Ross [78] proposed a new approach based on uniformization in 1987. Stiffness is managed by assuming the observation time intervals to be random variables with Erlangian distribution. Such a technique has also been called the external

uniformization [100]. The major difficulty with this approach is that it requires solving a large matrix linear system obtained from the state transition rate matrix [12]. Typically though, in biological systems, this state transition rate matrix is ultra sparse. Therefore well-developed sparse matrix linear system solvers are applicable. Numerical experiments presented here show that this approach is quite successful for several biological system models.

The remainder of this chapter is organized as follows: Section 2.2 illustrates the uniformization method and Ross' modified algorithm. Numerical results for several molecular biology models are given in Section 2.3, including comparisons between different methods. The final section concludes with a discussion of the capabilities of Ross' algorithm in the field of systems biology and suggests future research directions.

2.2 Methods and Algorithms

2.2.1 Mathematical Background of Master Equation

Let $\mathcal{X} = \{X(t), t \geq 0\}$ be a continuous time Markov chain (CTMC) with a state space $\mathcal{S}$. The number of possible states is finite and is equal to N. For any $i, j \in \mathcal{S}$, let $p_{ij}(t) = p\{X(t) = j \mid X(0) = i\}$. Given initial state probability vector $P(0)$, one is interested in computing $P(t)$, the state probability vector at time t. Obviously, $P(t) = P(0)\Pi(t)$, where $\Pi(t) = \big(p_{ij}(t)\big)$.

Suppose that when in state i, the CTMC makes a transition into state j at an instantaneous rate q_{ij} and let $q_i = \sum_{j \neq i} q_{ij}$ denote the rate at which $\mathcal{X}$ leaves state i. Then the state transition rate matrix

$$Q = \begin{bmatrix} -q_1 & q_{12} & \cdots & q_{1N} \\ q_{21} & -q_2 & \cdots & q_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N1} & q_{N2} & \cdots & -q_N \end{bmatrix}$$

is called the infinitesimal generator of $\mathcal{X}$. Since the state space $\mathcal{S}$ is finite, $\Pi(t)$ satisfies both Kolmogorov's backward equations $\Pi'(t) = Q\Pi(t)$ and Kolmogorov's forward equations $\Pi'(t) = \Pi(t)Q$, or in terms of $P(t)$,

$$P'(t) = P(t)Q. \tag{2.1}$$

Now, consider the chemical master equation. For reaction j,

$$x - \nu_j \overset{a_j(x-\nu_j)}{\longrightarrow} x. \tag{2.2}$$

The master equation

$$\frac{\partial p(x,t)}{\partial t} = \sum_{j=1}^{M} a_j(x-\nu_j)p(x-\nu_j,t) - a_j(x)p(x,t), \tag{2.3}$$

matches the matrix linear equation (2.1) [33].

The solution to (2.1) can be written as

$$P(t) = P(0)\Pi(t) = P(0)e^{Qt} = P(0)\sum_{n=0}^{\infty}\frac{(Qt)^n}{n!}. \tag{2.4}$$

However, using a truncation of the above infinite summation to approximate $P(t)$ is subject to severe truncation error [59].

2.2.2 Uniformization

Let $\eta = \max\limits_{1\le i\le N} q_i$. Now for any $\lambda \ge \eta$, consider an equivalent process where the instantaneous transition rate from any state i is λ and a $1 - q_i/\lambda$ fraction of these transitions returns back to state i immediately. This new process, called the uniformization of the original process, associates a discrete time Markov chain (DTMC) with a simple Poisson process (with rate λ). Algebraically, this is equivalent to

$$\Pi(t) = e^{Qt} = e^{-\lambda t}e^{\lambda t(I+\frac{Q}{\lambda})} = \sum_{n=0}^{\infty} e^{-\lambda t}\frac{(\lambda t)^n}{n!}\tilde{\Pi}^n, \tag{2.5}$$

where $\tilde{\Pi} = I + \frac{1}{\lambda}Q$ is the transition probability matrix for the DTMC. Hence,

$$P(t) = \sum_{n=0}^{\infty} e^{-\lambda t}\frac{(\lambda t)^n}{n!}v^{(n)}, \tag{2.6}$$

where $v^{(n)} = P(0)\tilde{\Pi}^n$ is the state probability vector of the DTMC after n transitions.

In practice, a partial sum of the infinite summation in (2.6) is used, which is also called the truncated uniformization method. For a given approximation error tolerance, the

(truncated) uniformization method without left side truncation (meaning that terms for small n in the infinite series are not dropped) requires $O(\eta t)$ terms. Even if the distribution is truncated from both sides (meaning that terms for both small and large n are dropped), it still requires $O(\sqrt{\eta t})$ terms. Additionally, successively squaring $\tilde{\Pi}$ to get the first significant DTMC state probability vector requires time $O(N^3 \log l)$, where l is the number of terms in the left tail $\sum_{n=0}^{l-1}$ of the infinite summation in (2.6) [72]. Hence, for large values of ηt and N, the computation can be cumbersome. For example, consider the bistable toggle switch problem described in Section 2.3.3. Let $\mathbf{Z}_{201}$ denote the integers modulo 201. If $\mathcal{S} = \mathbf{Z}_{201}^2$ and $t = 2.0 \times 10^5 s$, then $\eta \approx 0.4435$ and the value of $\lambda t = \eta t$ is approximately 8.9×10^4, which implies that at least 8.9×10^4 matrix vector multiplications are needed, while the dimension of the square matrix and vector is about 4.0×10^4.

However, the uniformization method has its own advantages. It is numerically stable and it only involves matrix vector multiplication, which makes it easy to implement. There is also a simple error bound derived from the Poisson distribution and the fact that $v^{(n)}$ is a probability vector [90].

2.2.3 External Uniformization Method

In 1987, Ross [78] introduced an external uniformization technique so that one could overcome the restriction on choosing λ larger than η. In the usual uniformization procedure, the CTMC is allowed to make transitions only at arrival epochs of a Poisson process, so λ has to be greater than η. In the external uniformization, instead, the CTMC is observed at arrival epochs of an independent Poisson process with any rate λ that is not related to η.

Formally, consider another random event $\mathcal{E}$, which occurs at times τ_1, τ_2, ..., where the intervals $\tau_k - \tau_{k-1}$, for $k > 1$ ($\tau_0 = 0$), are i.i.d. exponential random variables with rate λ independent of the Markov process $\mathcal{X}$. Assume the initial state $X(0) = i$. Then either a transition of $\mathcal{X}$ occurs before event $\mathcal{E}$, or $\mathcal{E}$ occurs first. Condition on which event occurs first and from the memoryless property of the exponential distribution,

$$p_{ij}(\tau_1) = \frac{q_i}{q_i + \lambda} \sum_{k \neq i} p_{kj}(\tau_1) \frac{q_{ik}}{q_i} + \frac{\lambda}{q_i + \lambda} \delta_{ij}, \tag{2.7}$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise. In matrix form this is equivalent to

$$\Pi(\tau_1) = \left(I - \frac{Q}{\lambda}\right)^{-1}. \tag{2.8}$$

From the Chapman-Kolmogorov equations

$$\Pi(\tau_r) = \left(I - \frac{Q}{\lambda}\right)^{-r}. \tag{2.9}$$

Now, use the fact that $E[\tau_r] = r/\lambda$ and $Var[\tau_r] = r/\lambda^2$. If $\lambda = r/t$, then the random variable τ_r approaches t as $r \to \infty$. As a consequence, $\Pi(\tau_r)$ should be a good approximation to $\Pi(t)$ [39]. To confirm this approximation, one just has to notice that

$$\lim_{r\to\infty} \Pi(\tau_r) = \lim_{r\to\infty} \left(I - \frac{Qt}{r}\right)^{-r} = e^{Qt} = \Pi(t). \tag{2.10}$$

There is an error bound for this method, based on the Taylor's expansion in [1],

$$\left|E[p_{ij}(\tau_r)] - p_{ij}(t)\right| \leq r^{-1}\left(\frac{t^2}{2}\right) \max_m \sum_k |q_{mk}q_{kj}|.$$

Another error bound based on the ergodic hypothesis of the original CTMC is in [93]. However, in his paper Ross provides some numerical evidence that this approach is accurate, even if a small number of steps r is chosen for the recursion. This is verified again in the numerical examples here.

Algorithm:

(i) Choose appropriate r according to the error bound and set $\lambda = r/t$.

(ii) Compute the matrix $(I - \frac{1}{\lambda}Q)$.

(iii) Set $P_r^{(0)} = P(0)$. Solve the matrix linear system of equations

$$P_r^{(i)}\left(I - \frac{1}{\lambda}Q\right) = P_r^{(i-1)} \tag{2.11}$$

recursively, for $i = 1, \ldots, r$. Then $P_r^{(i)}$ is an approximation for $P(it/r)$, and $P_r^{(r)}$ is the desired approximation for $P(t)$.

2.2.4 Uniformization Methods and Traditional ODE Methods

There is a close relationship between uniformization methods and traditional ODE methods. The original uniformization method is analogous to explicit ODE methods. The larger λt, which characterizes the stiffness of the system, is, the more terms in the infinite series need to be evaluated. On the other hand, Ross' modified uniformization method is analogous to implicit ODE methods, since it works for stiff systems at the expense of solving a matrix linear system. In fact, the linear system (2.11) can be rewritten as $(P_r^{(i)} - P_r^{(i-1)}) = (1/\lambda)P_r^{(i)}Q$. Now, recalling that $1/\lambda = t/r$, the modified uniformization method is indeed the implicit Euler ODE method. Similarly, from Equation (2.6), $v^{(n)} = v^{(n-1)}\tilde{\Pi} = v^{(n-1)}(I + Q/\lambda)$, which can be reformulated as $v^{(n)} - v^{(n-1)} = (1/\lambda)v^{(n-1)}Q$. This proves the connection between the original uniformization method and the explicit Euler ODE method.

Although the external uniformization method is only first order accurate as verified in numerical experiments, the numerical results are much better than expected even with large step sizes. This may have something to do with the stochastic derivation of the method. Essentially, each $P_r^{(i)}$ is the mean value of $P(\tau_i)$, where τ_i is a random variable with mean it/r and variance it^2/r^2. Also, as indicated in [78], this method, based on its stochastic meaning, could be used to calculate other stochastic properties like occupation times and the mean number of transitions. For instance, define $\bar{t}_{ij}(t)$ as the amount of time the CTMC starting from state i stays in state j by time t and the matrix $\bar{T}(t) = \big(\bar{t}_{ij}(t)\big)$. Then, it can be shown that $\bar{T}(\tau_r) = (\lambda I - Q)^{-r}$ approaches $\bar{T}(t)$ as $r \to \infty$, where $\lambda = r/t$, following similar reasoning as in Section 2.2.3 [78].

The stochastic approaches also provide insight into the problem, and suggest other possible numerical solution techniques, besides just explicit/implicit Euler methods, that would not be apparent from a purely ODE formulation. Here is an example from [100], which was originally proposed by Sumita and Shanthikumar [93]. Reference [100] proves that $\Pi(t) \to \hat{\Pi}^{t/\Delta}$ as $\Delta \to 0$, where $\hat{\Pi} = (\hat{p}_{ij})$ and

$$\hat{p}_{ij} = (1 - e^{-q_i\Delta})\frac{q_{ij}}{q_i} + e^{-q_i\Delta}\delta_{ij}. \tag{2.12}$$

From a stochastic point of view, $\hat{\Pi}^{t/\Delta}$ constructs the same CTMC as the original one, except that at most one transition is allowed during each time interval $(n\Delta, (n+1)\Delta]$ and any such transition is accounted for only at the end of each time interval. Similarly to the uniformization method, this explicit method computes a solution accurate within ϵ only if Δ is significantly small so that $1 - e^{-\eta\Delta} < \epsilon$, however, it imposes no further constraint on the choice of Δ. As for implicit methods, the reason why the external uniformization method is only first order accurate is that the random variable τ_i has a great deal of variability. To increase the accuracy, one would prefer τ_i to be a random variable with small variance, which would result in some better approximation scheme.

Another advantage of uniformization and its variants is that they automatically preserve certain properties of the numerical solution (the probability vector), such as the positivity and the conservation law of probability mass, because of their stochastic meaning.

2.3 Numerical Experiments

In this section the master equation of several models from molecular biology will be solved using the proposed method. Comparisons between the proposed method and some other methods are also given when appropriate. The performance of the proposed method depends largely on the effectiveness of the sparse matrix linear system solver. Hence, the algorithm has been implemented based on two different kinds of sparse matrix linear system solvers: iterative [81] (such as GMRES) and direct [17] (such as Gaussian elimination). The iterative solver software package chosen here is SPARSKIT [80], and the direct solver package used is SuiteSparse [16]. There is also a comparison between these two.

2.3.1 A Simple Birth-death Process

In this model x molecules are produced at a constant rate k and diminished at a rate proportional to the total number of molecules simultaneously. The reaction equations are

$$\left.\begin{array}{lcl} \emptyset & \xrightarrow{k} & x \\ x & \xrightarrow{\mu} & \emptyset \end{array}\right\}.$$

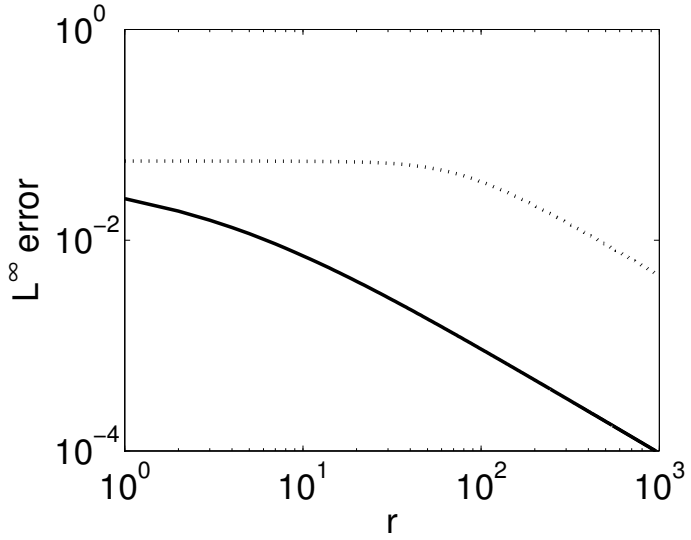


Figure 1. The L^∞ error versus r ($t_f = 100s$) using two different types of modified uniformization method. The solid line represents the approximation error using the matrix formed in (2.7) and the dotted line uses the approximation matrix in (2.12) with $\Delta = t/r$.

The master equation for this system is

$$\begin{aligned}\frac{\partial p(x,t)}{\partial t} &= kp(x-1,t) + \mu(x+1)p(x+1,t) \\ &\quad - (k+\mu x)p(x,t).\end{aligned} \tag{2.13}$$

This problem can be solved analytically if the initial data is given in the form of a Poisson distribution [47].

Figure 1 displays the computation error using the two algorithms proposed in Sections 2.2.3 and 2.2.4, with parameters $k = 1$, $\mu = 0.01$. It shows slow convergence of the explicit method (2.12), compared to the external uniformization method. Hence, the focus will be on the latter method in the following examples.

The solid line in Figure 1 indicates the first order accuracy of the external uniformization method. Such a nearly linear log-log plot bodes well for acceleration, such as Aitken's δ^2 method [92] and extrapolation methods. In Figure 2 the dotted line is the L^∞ error after applying Aitken's δ^2 method on $P_r^{(r)}$ and the dashed line is the L^∞ error after applying the same method on $P_{2^i}^{(2^i)}$, where $P_r^{(r)}$ is the final numerical result using the proposed method with r steps. Aitken's δ^2 method uses differences of consecutive terms, whose cancelation error is the source of the spikes in Figure 2, to accelerate the convergence of a given sequence.

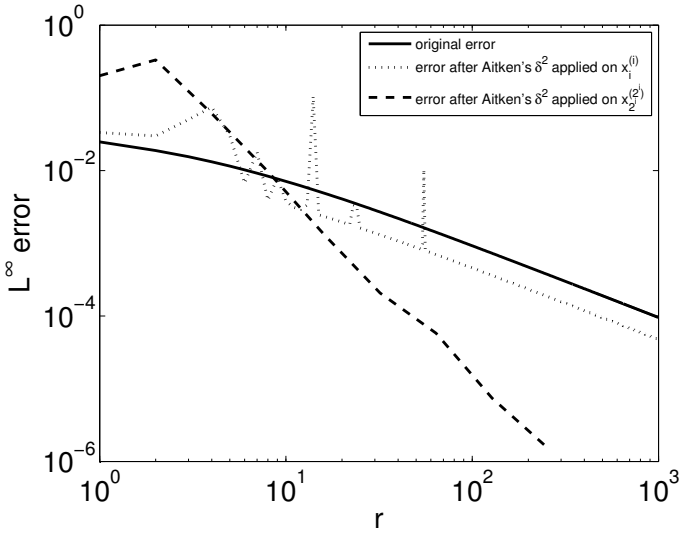


Figure 2. The L^∞ error versus r ($t_f = 100s$) using the external uniformization method before and after Aitken's δ^2 acceleration is applied.

Despite these spikes, Figure 2 shows improvements in error by Aitken's δ^2 method with no additional expensive computations. However, for such a small problem, it takes less than a second to run the algorithm even if $r = 10000$, which gives an L^∞ error less than 10^{-5} even without acceleration.

2.3.2 Schlögl Reaction

The Schlögl reaction [34], which is famous for its bistable distribution, is given by

$$\left.\begin{array}{rcl} b_1 + 2x & \underset{c_2}{\overset{c_1}{\rightleftharpoons}} & 3x \\ b_2 & \underset{c_4}{\overset{c_3}{\rightleftharpoons}} & x \end{array}\right\},$$

where b_1 and b_2 denote buffered species whose respective molecular populations are assumed to be constant. Schlögl reactions do not model any known real chemical system, since there are no actual trimolecular reactions like Schlögl reactions in nature, however, they are often used in numerical experiments.

Figure 3 compares the numerical results obtained from SSA and the external uniformization method. The parameters here are $c_1 = 3 \times 10^{-7}$, $c_2 = 10^{-4}$, $c_3 = 10^{-3}$, $c_4 = 3.5$, $b_1 = 1 \times 10^5$, $b_2 = 2 \times 10^5$, and the final time t_f is $4.0s$. This Schlögl model displays the

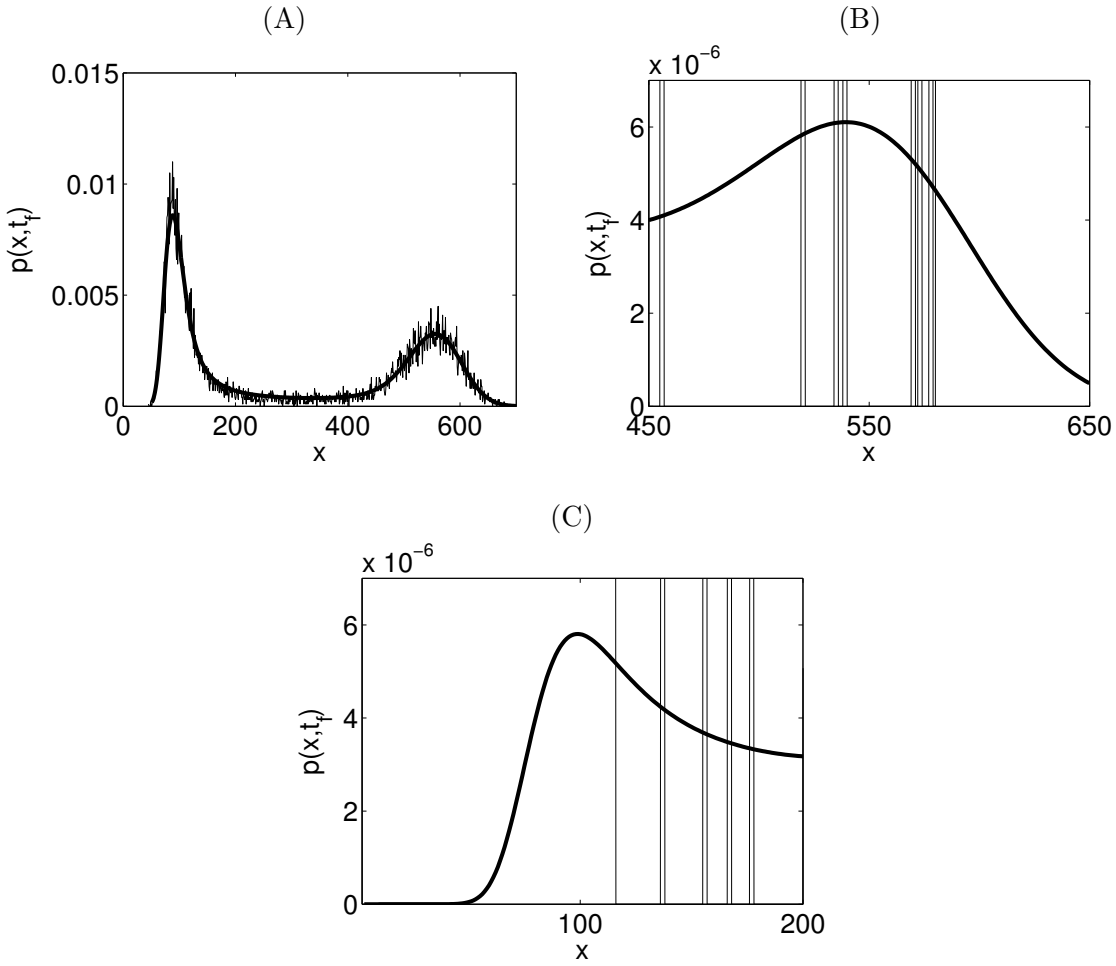


Figure 3. Comparison of the numerical results for the Schlögl reaction, with different initial states: (A) $x = 250$, (B) $x = 150$, (C) $x = 400$. The histogram (thin solid line) is based on 10,000 SSA simulations. The thick solid line is the numerical solution $P_{30}^{(30)}$ from the proposed method. (B) and (C) only display $p(x, t_f)$ around the stable state with less probability density.

bistable distribution only if the initial state is wisely chosen. In [34], the author has shown that this system has two stable states $x_1 = 82$ and $x_2 = 563$ and one barrier state $x_b = 248$. The bistable property of the distribution is apparent only when the initial state is close enough to the barrier state, as shown in Figure 3 (A). In Figure 3 (B), the initial state is set to the left of the barrier state. Trajectories starting from this initial state are more likely to end up around stable state x_1, which makes it more difficult to capture the other stable state x_2 as shown in the picture. Notice that 10,000 SSA simulations would only be able to predict probability mass larger than 10^{-4}, which is far greater than the peak value of $p(x, t_f)$

around x_2. Similarly, Figure 3 (C) illustrates what happens when the initial state is set to the right of the barrier — most trajectories end up around stable state x_2, which makes stable state x_1 more difficult to capture, especially so for Monte-Carlo algorithms. Figures 3 (B) and (C) show that the proposed method captures the behavior of the distribution around stable points with small probability density far better than histograms based on 10,000 SSA simulations.

Moreover, for such a small problem, the computational cost for the external uniformization method is negligible. For example, for $r = 30$, it takes approximately $0.018s$ if one uses a direct matrix linear system solver, or $0.37s$ if an iterative one is used instead. On the other hand, 10,000 simulation runs only produce a quite rough histogram estimation for the probability density function and already cost nearly two minutes on the same machine.

2.3.3 Toggle Switch Model

It has been proposed that gene regulatory networks with virtually any desired property can be constructed from simple regulatory elements. Examples of such properties include multistability and oscillations [30]. A genetic toggle switch, which has been constructed in *Escherichia coli* already, is such a simple regulatory element. It is a synthetic bistable gene regulatory network that could be constructed from any two repressible promoters. The bistability of the toggle switch model is obtained from the mutually inhibitory arrangement of these two [30]. However, any deterministic simulation of this model would only be able to predict at most one stable state, which makes the stochastic simulation crucial in this situation. The reaction equations of one such model are

$$\left.\begin{array}{lcl} \emptyset & \xrightarrow{\alpha/(\beta+x_2^2)} & x_1 \\ x_1 & \xrightarrow{\mu} & \emptyset \\ \emptyset & \xrightarrow{\gamma/(\delta+x_1^2)} & x_2 \\ x_2 & \xrightarrow{\mu} & \emptyset \end{array}\right\}$$

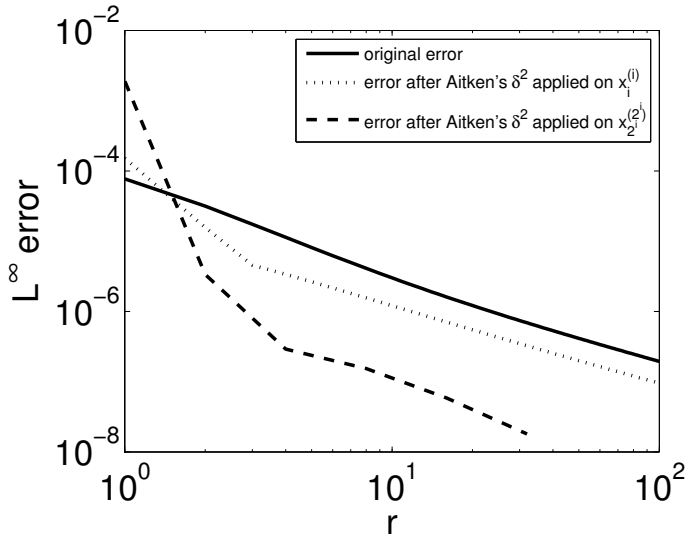


Figure 4. The L^∞ error versus r before and after Aitken's δ^2 acceleration is applied. The initial condition is $(x_1, x_2) = (60, 10)$ and the final time $t_f = 2.0 \times 10^5 s$.

with parameters $\alpha = \gamma = 1000, \beta = \delta = 6000$ and $\mu = 10^{-3}$ [21].

In references [21] and [40] the toggle switch problem is solved with different formulations. In [21] the same problem is solved on a state space $\mathbf{Z}^2_{201}$, the integer lattice points in $[0, 200] \times [0, 200]$. The master equation is approximated by an ODE system of dimension 400. The ODE system is then solved in Matlab (ode15s). Only solutions and errors in different norms are reported in [21]. In [40] the problem setting is a little different with a state space $\mathbf{Z}^2_{51}$. A sparse grid technique is used to make the state space even smaller. The exponentials of the smaller matrix are then computed by some Krylov space projection methods [88]. As reported in [40], it takes less than 15 minutes to solve another problem with $16^{10} \approx 10^{12}$ states (using a matrix of dimension 1001×16 after approximation) on a 2 GHz AMD64 based PC with 1 GByte of memory. Note that both these methods are essentially approximation techniques applied on the state space. Furthermore, the latter sparse grid technique can also be used as an approximation technique in conjunction with the algorithm proposed here. The master equation for the toggle switch model can also be solved by Monte Carlo algorithms indirectly. Their numerical results (implemented in StochKit [52]) are listed in Figure 5 and Table 1.

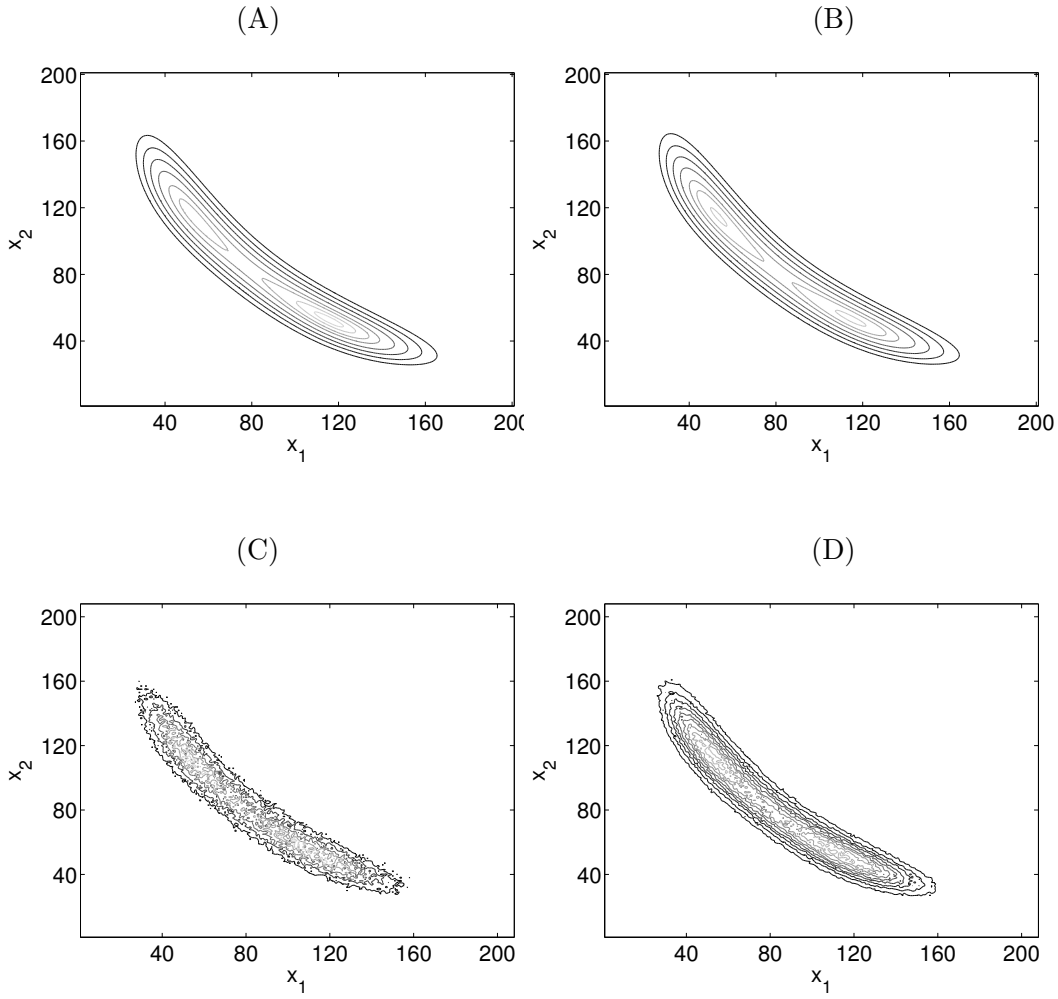


Figure 5. Toggle switch computational results. The initial condition is $(x_1, x_2) = (60, 10)$ and the final time t_f is $2.0 \times 10^5 s$. (A) Contour plot for $P_{20}^{(10)}$. (B) Contour plot for $P_{20}^{(20)}$. (C) Contour plot based on 100,000 SSA simulations. (D) Contour plot based on 1,000,000 SSA simulations.

Figure 4 shows the computation error for different r values using the proposed algorithm. The L^∞ norm error is estimated using a reference solution ($r = 10000$) and the results are similar to those in Figure 2.

Figure 5 contains the contour plots for numerical results. The bistable property is clearly apparent in each contour plot. Again, the contour plots obtained from SSA have much more noise than those two from the proposed method. Running 1,000,000 SSA simulations takes more than 10 hours, while the proposed method just takes a few seconds on the same machine. Figure 5 also contains a contour plot of $P_{20}^{(10)}$, which approximates the probability

(A)

	SSA	Adaptive τ-leaping
10^6 runs	52670	49290

(B)

r	BICGSTAB	BICGSTAB+ILUT	UMFPACK
5	21.3	6.2	1.3
10	31.8	10.5	1.6
20	40.5	15.5	2.3
40	50.4	21.0	3.6
80	59.1	29.6	6.1

Table 1. CPU time (sec) for different methods. (A) Monte Carlo algorithms. The error control epsilon is set at 0.03 in the adaptive τ-leaping method. (B) The proposed method using different r and matrix linear system solvers.

density function at time $1.0 \times 10^5 s$. Actually, the vectors $P_{20}^{(i)}$ $(i = 1, ..., 20)$ approximate the probability density function at any epoch $(i/20) \times 2.0 \times 10^5 s$. These probability density functions altogether provide information on how the system reaches the equilibrium state.

Table 1 lists the CPU time for two different methods, including a comparison between iterative and direct matrix linear system solvers with different r values for the proposed method. It shows that for this model problem the direct solver (UMFPACK, the unsymmetric multifrontal method for sparse LU factorization package) performs better than the iterative solver (BICGSTAB, the biconjugate gradient stabilized method) with or without preconditioners (ILUT, the incomplete LU factorization with dual truncation strategy) and all of them outperform Monte-Carlo algorithms.

2.3.4 A Prototypical Cell Cycle Model

A simple deterministic cell cycle model can be described by the normalized phenomenological rate equations [101, 94]:

$$\begin{aligned}
\frac{d}{dt}Y_1 &= \kappa_1 m - (\kappa_2' + \kappa_2'' Y_2) Y_1, \\
\frac{d}{dt}Y_2 &= \frac{\kappa_3'' Y_3 (1 - Y_2)}{\Gamma_3 + (1 - Y_2)} - \frac{\kappa_4 Y_1 Y_2}{\Gamma_4 + Y_2}, \\
\frac{d}{dt}Y_3 &= \kappa_5' + \kappa_5'' \frac{Y_1^2}{\Gamma_5^2 + Y_1^2} - \kappa_6 Y_3.
\end{aligned}$$

κ_1	k_1/c_{CycB}	$0.01\,\mathrm{min}^{-1}$
κ_2'	k_2'	$0.04\,\mathrm{min}^{-1}$
κ_2''	$k_2''c_{Cdh1}$	$1.0\,\mathrm{min}^{-1}$
κ_3''	$k_3''c_{Cdc20}/c_{Cdh1}$	$10.0\,\mathrm{min}^{-1}$
κ_4	k_4c_{CycB}/c_{Cdh1}	$35\,\mathrm{min}^{-1}$
κ_5'	k_5'/c_{Cdc20}	$0.005\,\mathrm{min}^{-1}$
κ_5''	k_5''/c_{Cdc20}	$0.2\,\mathrm{min}^{-1}$
κ_6	k_6	$0.1\,\mathrm{min}^{-1}$
Γ_3	J_3/c_{Cdh1}	0.04
Γ_4	J_4/c_{Cdh1}	0.04
Γ_5	J_5/c_{CycB}	0.3

Table 2. Relationships between the normalized and unnormalized parameters and the values of the normalized parameters.

In the above equations, $Y = (X_{CycB}, X_{Cdh1}, X_{Cdc20})$ and $X_S = [S]/c_S$ is the normalized concentration of species S and c_S is the characteristic concentration of the species. The equations also assume that the normalized concentration of total Cdh1 is 1, so that the concentration of the phosphorylated form can be written as $X_{Cdh1P} = 1 - X_{Cdh1}$. The variable m reflects the fact that $CycB$ is assumed synthesized at a supralinear rate and thus its concentration increases with cell mass. In terms of numbers of molecules $y = (x_{CycB}, x_{Cdh1}, x_{Cdc20})$, where $x_S = c_S X_S V_s$ (V_s is the nominal volume of the cell times Avogadro's number and equals 18 molecules/nMolar here), the equations are:

$$\begin{aligned}
\frac{d}{dt}y_1 &= k_1 m V_s - (k_2' + \frac{k_2''}{V_s}y_2)y_1, \\
\frac{d}{dt}y_2 &= \frac{k_3'' y_3 (c_{Cdh1}V_s - y_2)}{J_3 V_s + (c_{Cdh1}V_s - y_2)} - \frac{k_4 y_1 y_2}{J_4 V_s + y_2}, \\
\frac{d}{dt}y_3 &= k_5' V_s + k_5'' V_s \frac{y_1^2}{(J_5 V_s)^2 + y_1^2} - k_6 y_3.
\end{aligned} \tag{2.14}$$

The relationships between the normalized and unnormalized parameters, as well as the values of the normalized parameters, are given in Table 2.

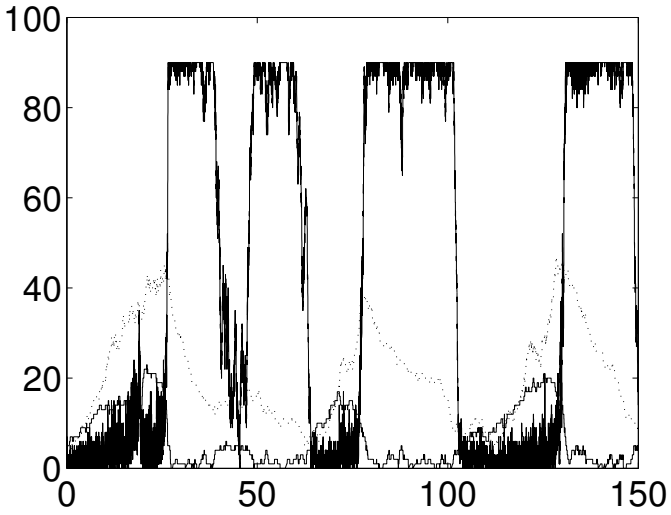


Figure 6. One trajectory for the cell cycle model ($t_f = 150\,\text{min}$). CycB, grey line. Cdh1, solid line. Cdc20, dotted line.

In order to get an accurate stochastic model of this system, Equation (2.14) needs to be unpacked into elementary chemical reactions without intermediates and with variable propensities.

$$\left.\begin{array}{rcl}
\emptyset & \underset{k_2'}{\overset{k_1 m V_s}{\rightleftharpoons}} & x_{CycB} \\
x_{CycB} + x_{Cdh1} & \overset{k_2''/V_s}{\longrightarrow} & x_{Cdh1} \\
x_{CycB} + x_{Cdh1} & \overset{\frac{k_4}{J_4 V_s + x_{Cdh1}}}{\longrightarrow} & x_{CycB} + x_{Cdh1P} \\
x_{Cdc20} + x_{Cdh1P} & \overset{\frac{k_3''}{J_3 V_s + x_{Cdh1P}}}{\longrightarrow} & x_{Cdc20} + x_{Cdh1} \\
\emptyset & \underset{k_6}{\overset{k_5' V_s + k_5'' V_s \frac{x_{CycB}^2}{(J_5 V_s)^2 + x_{CycB}^2}}{\rightleftharpoons}} & x_{Cdc20}
\end{array}\right\}.$$

In the numerical experiments the characteristic concentrations $c_{CycB} = 5.0nM$, $c_{Cdh1} = 5.0nM$, $c_{Cdc20} = 5.0nM$. The system is initialized with $x = (x_{CycB}, x_{Cdh1}, x_{Cdc20}) = (5, 5, 5)$ and $m = 1.5$. Figure 6 and Figure 7 display the numerical results, while Table 3 lists the computational costs. The numerical results from the proposed method match the histograms obtained from SSA simulations and are much smoother. As for the computational costs, in this example it is the iterative solver with preconditioner that performs the best, at about half the cost of the Monte Carlo methods. Notice that since the matrix Q is singular, the

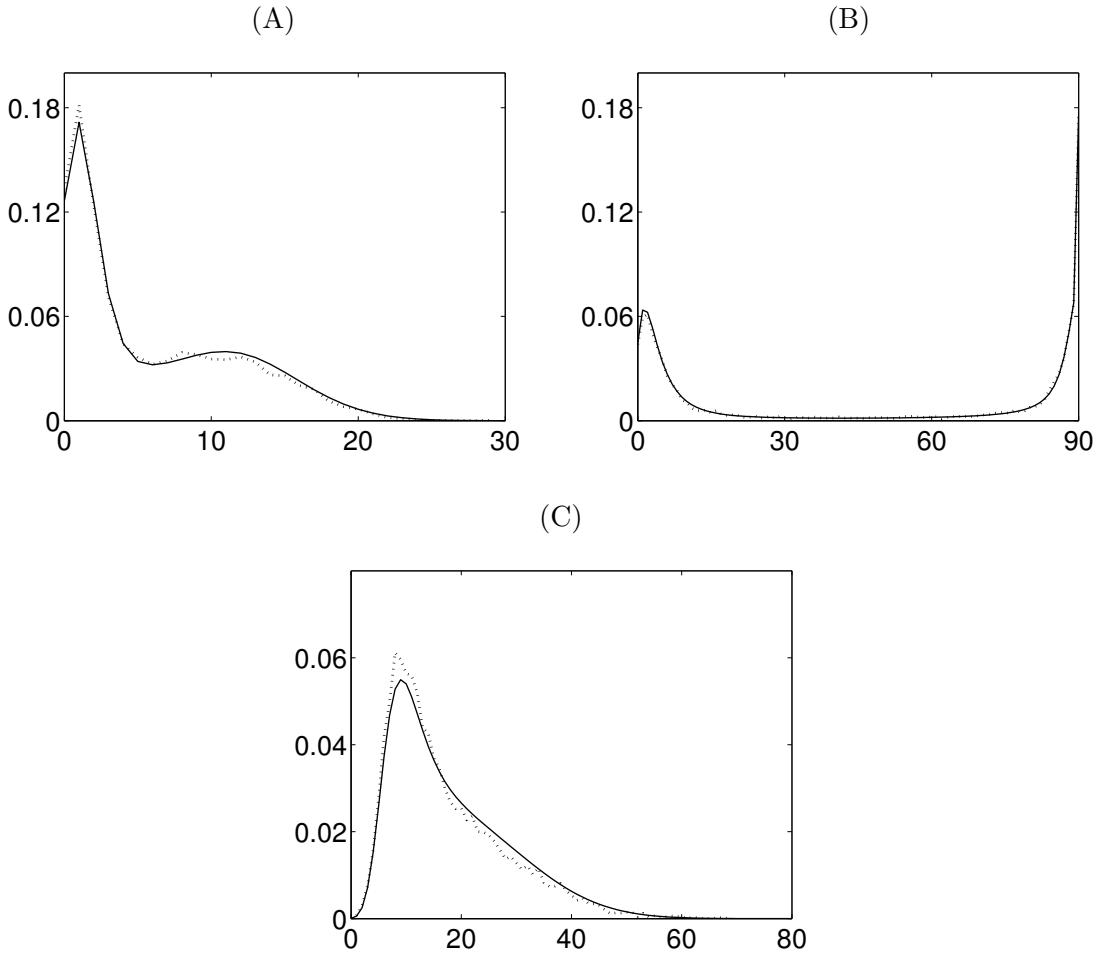


Figure 7. Cell cycle model computational results. Comparisons of estimations for marginal probability distributions. The histograms (dotted lines) are based on 10,000 SSA simulations. The solid lines are based on $P_{10}^{(10)}$ computed from the proposed method. The final time is 100 min. (A) CycB. (B) Cdh1. (C) Cdc20.

(A)

	SSA	Adaptive τ-leaping
10^4 runs	251.9	257.9

(B)

r	BICGSTAB	BICGSTAB+ILUT	UMFPACK
10	62670.6	108.6	517.0
20	4082.8	147.2	590.5

Table 3. CPU time (sec) for different methods. (A) Monte Carlo algorithms (StochKit). The error control epsilon in the adaptive τ-leaping method equals 0.03. (B) The proposed method using different matrix linear system solvers and different r.

matrix $(I - Qt/r)$ becomes ill conditioned as t increases (or as r decreases), which may explain why it takes iterative solvers without preconditioning more CPU time to solve the problem at a smaller r. However, this ill conditioning has less impact on preconditioned iterative solvers.

2.4 Conclusions

In the theory of Markov processes, determining the value of the state probability vector at any time before the system reaches the stable state is called transient analysis. In contrast to steady state analysis, transient analysis requires solving linear differential equations instead of linear algebraic equations, which makes transient analysis much more difficult. Nevertheless, many methods have been proposed for transient analysis, based on traditional ODE solvers, the exponential of a matrix, Laplace transforms, Krylov subspaces, and uniformization [39, 86]. The uniformization method was proposed by Jensen and has become very popular in the last twenty years. This chapter has mainly investigated a variant of the standard uniformization method, called the external uniformization method. Numerical results here show that for a number of problems, especially when the problem size is small, the external uniformization method is both numerically efficient and accurate.

One important feature of (external) uniformization method is its simplicity. Only an efficient sparse matrix linear system solver is needed. When the problem size is small, direct linear system solvers usually outperform iterative solvers, but this may not be the case for higher dimension problems or problems with more irregular structures. Generally, the performance of iterative linear solvers depends crucially on preconditioning as shown in the last model problem.

Like other direct methods for the chemical master equations, the major computational challenge for the external uniformization method comes from the curse of dimensionality. Combining the method with model reduction techniques, like sparse grid approximation for the state space, and future novel techniques for efficiently computing $P_r^{(r)}$, holds promise.

Chapter 3: ANALYSIS OF THE GK SWITCH USING THE CME ON PARALLEL MACHINES

3.1 Introduction

Consider a simple common enzyme kinetics reaction:

$$E + S \underset{k_{-1e}}{\overset{k_{1e}}{\rightleftharpoons}} E : S \overset{k_{2e}}{\rightarrow} S_p + E.$$

Here substrate S is converted to product S_p enzymatically by the enzyme E. The rate equations corresponding to this reaction are

$$\begin{aligned}
\frac{d[S]}{dt} &= -k_{1e}[S][E] + k_{-1e}[E:S], \\
\frac{d[E:S]}{dt} &= -(k_{-1e} + k_{2e})[E:S] + k_{1e}[S][E], \\
\frac{d[E]}{dt} &= -k_{1e}[S][E] + (k_{-1e} + k_{2e})[E:S],
\end{aligned}$$

where $[X]$ represents the concentration of species X.

In most cases, the timescale of substrate S is the timescale of interest. On this timescale, the (constant) total enzyme concentration $E_T = [E] + [E:S]$ is the slow variable, and the concentration of the enzyme-substrate complex $[E:S]$ is the fast variable, which is assumed to reach a steady state quickly. Mathematically, this means

$$[E] + [E:S] = E_T = \text{constant},$$

$$-\frac{d[E]}{dt} = \frac{d[E:S]}{dt} = -(k_{-1e} + k_{2e})[E:S] + k_{1e}[S][E] \approx 0.$$

Taking this latter rate as exactly zero, it is possible to simplify the system to one deterministic equation only involving S,

$$\frac{d[S]}{dt} = -k_{2e}[E:S] = -\frac{k_{2e}E_T[S]}{k_{me} + [S]},$$

where $k_{me} = (k_{-1e} + k_{2e})/k_{1e}$. This approach, called quasi-steady state approximation (QSSA) [70], is widely used to simulate the enzyme kinetics reaction approximately. The condition for QSSA to be valid here is that $E_T \ll S_0 + k_{me}$, where S_0 is the initial condition

of $[S]$. Conditions like this are also called Michaelis-Menten (MM) conditions [57]. There are cases, like protein interaction networks, where such MM conditions can not hold. For example, in protein interaction networks one enzyme may have multiple substrates and one substrate may be acted upon by multiple enzymes. Sometimes enzymes and substrates may even exchange roles, in which case conditions like $E_T \ll S_0 + k_{me}$ will clearly not be true.

The Goldbeter-Koshland switch [37] is a coupled enzyme kinetics system. It is composed of a substrate-product pair (S and S_p). This substrate-product pair is interconverted by two enzymes (D and E):

$$D + S_p \underset{k_{-1d}}{\overset{k_{1d}}{\rightleftharpoons}} D : S_p \overset{k_{2d}}{\rightarrow} S + D,$$

$$E + S \underset{k_{-1e}}{\overset{k_{1e}}{\rightleftharpoons}} E : S \overset{k_{2e}}{\rightarrow} S_p + E.$$

Based on the QSSA assumption, one single dynamical equation can be used to characterize the deterministic time evolution of the system, which is the well known sigmoidal Goldbeter-Koshland function [14]. Their analysis is fine as long as the MM condition holds. If the MM condition is violated, a modified QSSA approach, tQSSA, may be used. However, the tQSSA approach also has its own constraints.

These deterministic approaches are only useful if the continuum point of view is valid for describing the chemical system. Random fluctuations come into play when the number of molecules is on the order of hundreds, where stochastic approaches are necessary. One common applicable approach is Monte Carlo methods, such as the stochastic simulation algorithm (SSA) [32, 31] and τ-leaping [35]. There are papers trying to combine Monte Carlo methods with different QSSA/tQSSA assumptions [70, 11]. Another well-known approach is solving the chemical master equation. One benefit of using the chemical master equation is that it gives the time evolution of the probability distribution of all possible states directly. However, this approach is computationally intensive. This chapter offers some insight into how to solve the chemical master equation numerically on the Goldbeter-Koshland model. Numerical results will also be discussed.

3.2 Markov Processes and the CME

One common assumption often made about a chemical system is the Markov assumption [95, 33], which roughly means there is no dependence of the future on the past when the present is given. Under proper conditions, ordinary differential equations like the Kolmogorov backward/forward equations [77] can be used to characterize systems satisfying the Markov property. If the probability space is finite, then the Kolmogorov backward/forward equations can be reduced to one equation, which coincides with the chemical master equation

$$\frac{dP(t)}{dt} = P(t)Q, \tag{3.1}$$

where $P(t)$ is the probability/stochastic (row) vector for all possible states and

$$Q = \begin{bmatrix} -q_1 & q_{12} & \cdots & q_{1N} \\ q_{21} & -q_2 & \cdots & q_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N1} & q_{N2} & \cdots & -q_N \end{bmatrix} \tag{3.2}$$

is the infinitesimal generator matrix. N is the number of possible states. Each component q_{ij} $(i \neq j)$ denotes the nonnegative instantaneous constant rate at which the system makes one transition from state i into state j in an infinitesimal time interval. To satisfy the conservation law of overall probability mass, each row of the matrix Q sums up to zero, i.e., $q_i = \sum_{j \neq i} q_{ij}$.

Now, the solution to the master equation can be written as

$$P(t) = P(0)e^{Qt} = P(0)\sum_{n=0}^{\infty} \frac{(Qt)^n}{n!}. \tag{3.3}$$

However, using a truncation of the above infinite summation to approximate $P(t)$ may subject to severe truncation error [59].

The system reaches its steady state when its probability distribution approaches a constant over time. In other words, when $dP(t)/dt = 0$. Hence, the steady state probability distribution P_s satisfies $P_sQ = 0$. Common numerical methods to compute P_s are discussed in Section 3.4.

The size of the matrix Q in the master equation for the GK switch and many other chemical systems is often huge, on the order of 10^5 to 10^9. On the other hand, it is often extremely sparse. Each its components q_{ij} is nonzero only if there is one reaction channel that leads the system to make a change from state i into state j. There are six different reaction channels in the GK switch, which implies that there are at most seven nonzero elements in each row of the matrix Q. This special property of the matrix Q will be fully exploited when designing an efficient numerical algorithm to evaluate e^{Qt} or to solve the linear system $P_s Q = 0$.

3.3 Transient Analysis of Markov Chains

3.3.1 Numerical Methods for the Transient Analysis

Computing $P(t)$ from a chemical master equation is also called numerical transient analysis of a Markov model [72], since $P(t)$ fully characterizes the transient behavior of the Markov model. A large amount of research has been conducted in this area. For dealing with large, continuous-time, discrete-state Markov chains, there are three basic categories of methods [90]. The first one consists of methods specifically designed for Markov chains, basically, the uniformization/randomization method and its variants. The second one is composed of traditional numerical ODE solvers, single step and multi-step. The last one is called Krylov space methods.

The uniformization method was first proposed by Jensen [48] in 1953. It offers a stable way to evaluate the infinite series from the matrix exponential, and is acclaimed as one of the most efficient methods available to find transient solutions of Markov models. Unfortunately, it is not efficient when the ODE system is stiff, which is often the case in biological systems. In 1989, Ross [78] introduced a modified uniformization technique to handle stiff systems. Algebraically, the essential part of the original uniformization method is the transformation

$$e^{Qt} = e^{-\lambda t} e^{\lambda t (I + \frac{Q}{\lambda})} = \sum_{n=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \tilde{\Pi}^n, \tag{3.4}$$

where $\lambda \geq q_{\max} = \max_{1 \leq i \leq N} q_i$. Thus, $\tilde{\Pi} = I + Q/\lambda$ becomes a probability matrix, so the multiplication between $P(0)$ and $\tilde{\Pi}^n$ is now stable. In analogy to ODE solvers, the original uniformization method is similar to an explicit method. A larger λ makes the method more stable, but it also means more terms in the infinite series need to be evaluated. As with ODE solvers, one way to address this issue is to choose λ adaptively, resulting in the adaptive uniformization method [96]. Another way is to use implicit methods, which is exactly what has been proposed in Ross' paper. A probability flavored derivation of this modified uniformization method is given in Ross' paper, whereas algebraically, the basic idea is to use the limit equality

$$\lim_{r \to \infty} \left(I - \frac{Qt}{r} \right)^{-r} = e^{Qt} \tag{3.5}$$

to approximate the matrix exponential. The algorithm is to first choose an appropriate r and set $\lambda = r/t$ and $P_r^{(0)} = P(0)$. Then solve the linear system $P_r^{(i)}(I - Q/\lambda) = P_r^{(i-1)}$ recursively, for $i = 1, ..., r$. Now $P_r^{(i)}$ is an approximation for $P(it/r)$, and $P_r^{(r)}$ is the desired approximation for $P(t)$. Rewrite the linear system in a different form,

$$(P_r^{(i)} - P_r^{(i-1)}) = \frac{1}{\lambda} P_r^{(i)} Q. \tag{3.6}$$

Now, considering that $1/\lambda = t/r$, the modified uniformization method is indeed the implicit Euler ODE method. This also explains why the modified method works for stiff systems at the expense of solving a linear system.

Now consider the second category of methods, general ODE algorithms. Only absolutely stable methods will be discussed here. It is well known that all absolutely stable linear multi-step ODE methods are implicit and none of them has an error order greater than two. The first order implicit Euler method has been discussed above. Another choice is the trapezoidal method, which is the absolutely stable linear multi-step ODE method that has the least local truncation error. The algorithm is similar to the modified uniformization method, except that a different linear system, $P_r^{(i)}\big(I - Q/(2\lambda)\big) = P_r^{(i-1)}\big(I + Q/(2\lambda)\big)$, is solved recursively this time.

In recent years, Krylov space methods have been introduced to many numerical computation domains including computing matrix exponentials. To compute matrix exponentials like $P(0)e^{Qt}$, the idea is to project e^{Qt} onto some Krylov space $K_m\big(Q, P(0)\big)$. However, it can be shown that such a projection can be replaced with the exponential of the restriction of Qt onto the same Krylov space. By doing so, the large sparse problem is now replaced with a small dense problem, computing matrix exponentials of upper Hessenberg matrices, which is usually done by Padé approximation. In reality, the projection is not computed in one shot. On the contrary, a time-stepping strategy in t along with an error estimation is embedded within the process [59].

3.3.2 Numerical Experiments

In this section, the master equation the GK switch will be solved using methods described in the previous section. Comparisons between these methods and SSA are also given when appropriate. Since the performance of implicit ODE methods largely depends on the effectiveness of the linear system solver, the algorithm has been implemented using several different kinds of sparse linear system solvers, including their parallel versions.

The parameter values considered here are $k_{1d} = 0.05555\,\mathrm{min}^{-1}$, $k_{-1d} = 0.83\,\mathrm{min}^{-1}$, $k_{2d} = 0.17\,\mathrm{min}^{-1}$, $k_{1e} = 0.05\,\mathrm{min}^{-1}$, $k_{-1e} = 0.8\,\mathrm{min}^{-1}$ and $k_{2e} = 0.1\,\mathrm{min}^{-1}$. The problem is first posed on a relatively small state space, $(S_T, D_T, E_T) = (900, 9, 5)$, with initial state condition $(S_p, D : S_p, E : S) = (900, 0, 0)$ and the final time $t_f = 100\,\mathrm{min}$.

Figure 8 compares the numerical results from the implicit Euler method and the trapezoidal method ($r = 100$) and Krylov space method ($m = 60$) along with the histogram generated by 50,000 SSA runs. It shows a clear match when estimating the probability density function for species $D : S_p$ and $E : S$. However, for species S_p, the implicit Euler method fails to match the others, whereas the trapezoidal method gives a better, smooth estimation around the peak but a worse, oscillatory estimation near the boundary. The

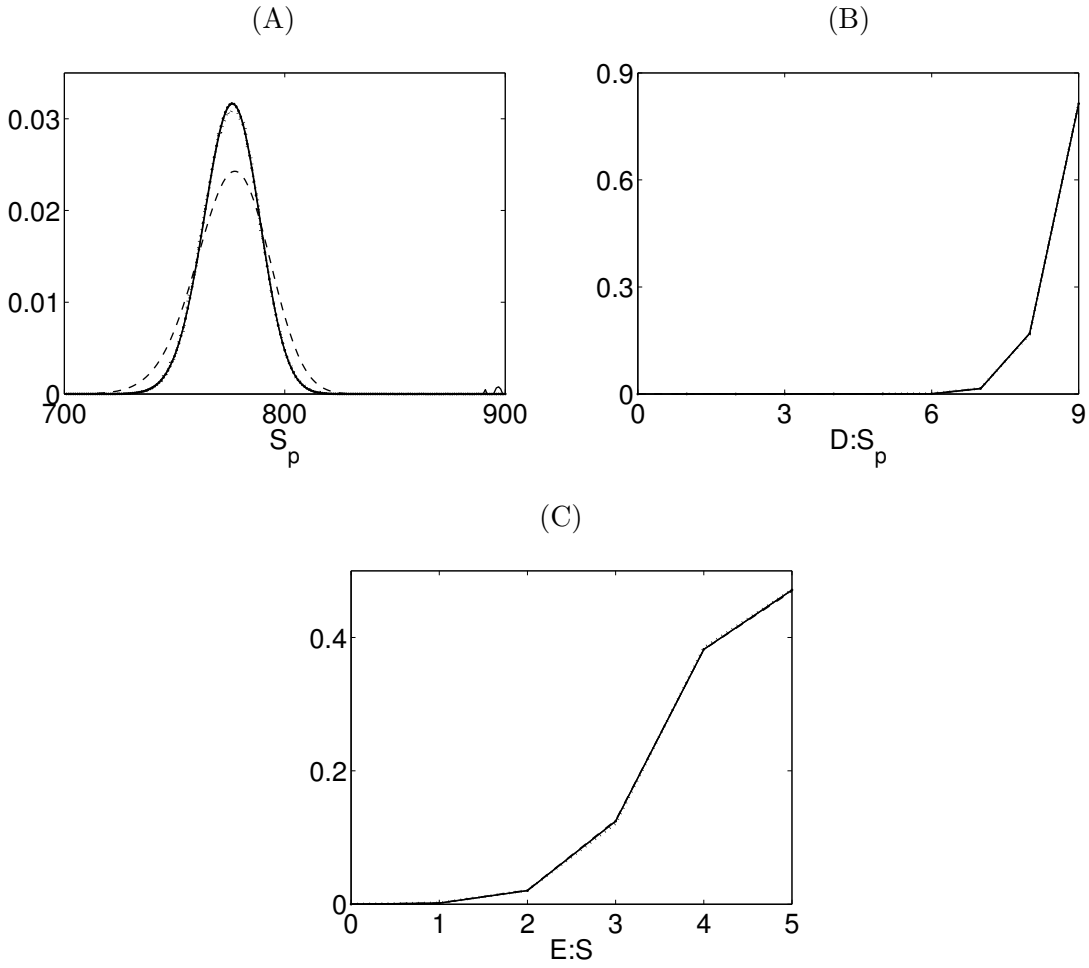


Figure 8. Numerical estimations for the probability distributions after 100 min. Histogram from SSA, dotted line. Implicit Euler method, dashed line. Trapezoidal method, solid line. Krylov space method, dash-dot line.

numerical results from the Krylov space method are the best of all, yet it costs the most computation time.

Under the same problem setting, the final time t_f is now extended from 100 min to 1000 min to examine the time scalability of these algorithms. The time step is fixed at 1 min in the implicit Euler method and the trapezoidal method. Figure 9 shows deterministic time courses of the network, which indicates that the system is near stable state at time $t_f = 1000\,\text{min}$ when $(S_T, D_T, E_T) = (900, 9, 5)$.

Two different kinds of sparse linear solvers were tried here to solve the linear system imposed by implicit ODE algorithms. One linear solver is a direct sparse solver [17], like Gaussian elimination, which fits well situations like this. Since the matrix in the linear

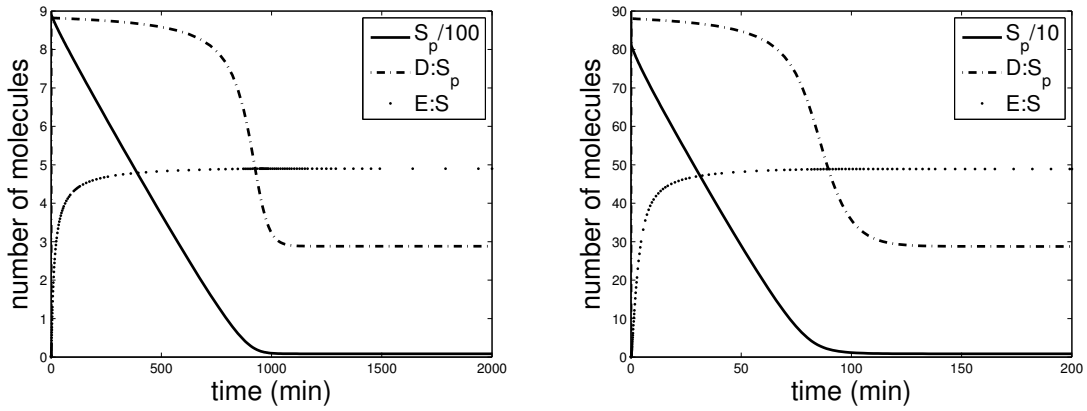


Figure 9. The deterministic time course of the network. Top: $(S_T, D_T, E_T) = (900, 9, 5)$. Bottom: $(S_T, D_T, E_T) = (900, 90, 50)$).

system never changes from step to step, only one PLU factorization, which is the most computational intensive part, is actually needed. However, PLU factorization requires extra storage space, which would be an issue when the matrix size grows. The software package used here is SuiteSparse [16] (UMFPACK, the unsymmetric multifrontal method for sparse LU factorization). The other linear solver is an iterative solver [81], like GMRES, which is implemented here combined with properly chosen preconditioners. The software package used here is SPARSKIT [80] (BiCGSTAB, the biconjugate gradient stabilized method, combined with ILUT, the incomplete LU factorization with dual truncation strategy, as preconditioner). Like the PLU factorization for direct solvers, the preconditioner for iterative solvers is computed once and for all.

t_f	SSA	Trapezoidal		Krylov
		Direct	Iterative	
100	100.8	20.3	148.2	621.8
200	210.2	32.9	202.1	1330.6
400	430.8	58.3	270.0	1604.5
600	646.5	83.6	334.2	1749.7
800	860.6	108.8	387.6	1885.2
1000	1029.5	134.2	433.2	2008.6

Table 4. CPU time (sec) for different algorithms. $(S_T, D_T, E_T) = (900, 9, 5)$.

In terms of CPU time, Table 4 shows that after the system evolves through some time span, implicit ODE algorithms and Krylov space methods will eventually have at least the

same time-scalability as SSA. There are cases where ODE algorithms and Krylov space methods have much better scalability than Monte-Carlo algorithms, which usually happens when the system is near stable state. At that point, the current state probability distribution $P_r^{(i)}$ differs only little from the previous state probability distribution $P_r^{(i-1)}$, so only a few iterations per step are needed.

Next, the state space size is expanded to examine the space-scalability of these algorithms in terms of CPU time for the same problem. Four problems with different state space sizes are compared here in Table 5. Figure 10 compares the numerical results at space size $(S_T, D_T, E_T) = (900, 90, 50)$ and the result is similar to Figure 8.

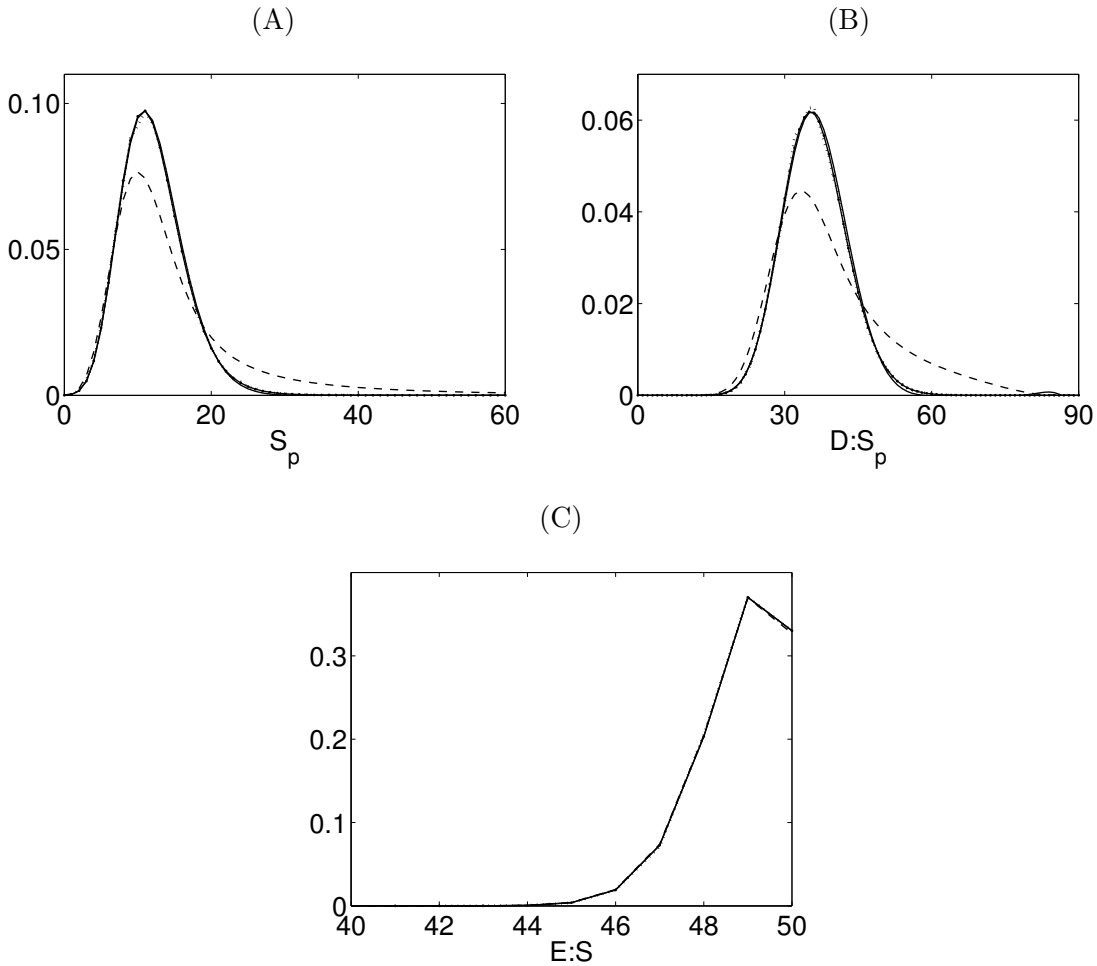


Figure 10. Numerical estimations for the probability distributions after 100 min. Histogram from SSA, dotted line. Implicit Euler method, dashed line. Trapezoidal method, solid line. Krylov space method, dash-dot line.

D_T, E_T	SSA	Trapezoidal		Krylov
		Direct	Iterative	
9,5	100.8	20.3	148.2	621.8
9,50	174.9	1416.1	1874.2	4937.4
90,5	500.6	863.4	890.4	42956.6
90,50	1006.3	*	12091.2	277798.6

* Out of memory.

Table 5. CPU time (sec) for different algorithms. S_T is fixed at 900. The final time t_f and the number of SSA runs are fixed at 100 min and 50,000, respectively.

As shown in Table 5, ODE algorithms and Krylov space methods have worse space-scalability than Monte-Carlo algorithms, which is expected.

Finally, the problem is solved on a parallel machine, with a state space $(S_T, D_T, E_T) = (900, 90, 5)$. The software package used here is pARMS [53] (FGMRESD, the distributed version of flexible GMRES, as solver, combined with various preconditioners). pARMS uses ARMS, an algebraic recursive multilevel solver, rooted in multi-level ILU type techniques as the building block. There are three ways to use ARMS in a parallel environment. The first and simplest way is to use an additive Schwarz procedure in which ARMS is used as preconditioner for the local solver (`add_arms`). The second and third approaches rely on a Schur complement-type technique. The Schur complement relates to equations associated with interface points, where internal points are implicitly used as intermediate variables. In the second approach (`lsch_arms`) the ARMS reordering is applied locally, while in the third approach (`sch_sgs`, `sch_gilu0`) it is applied globally. A preconditioner's name may be followed by "`no its`" and/or "`ovp`", meaning the preconditioner is applied without inner iterations and/or has overlapping, respectively. The partition of the matrix is done by the sequential algorithm METIS developed by Kumar and Karypis [49]. Figure 11 shows the results.

Among all the five preconditioning methods used here, `sch_gilu0` consumes the most CPU time, yet it also has the best parallel efficiency up to 16 processor nodes.

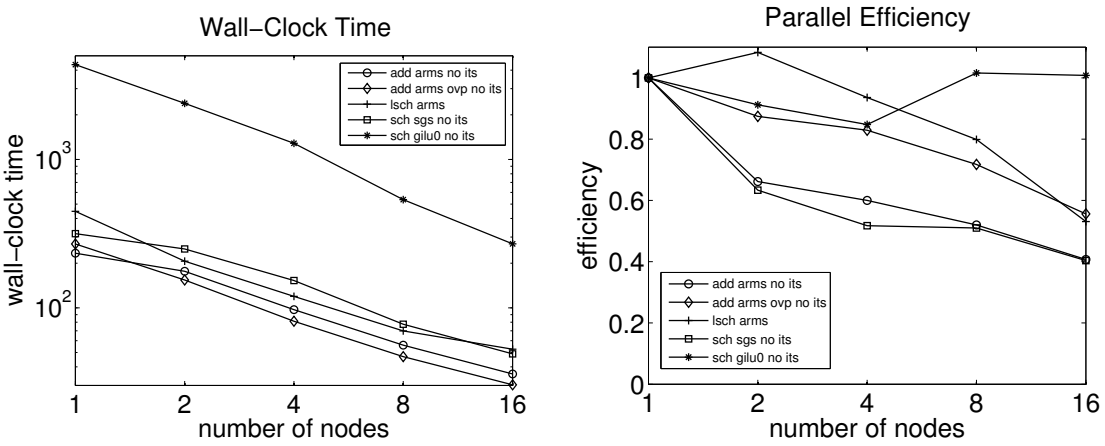


Figure 11. Solution times (top) and parallel efficiency (bottom) for a GK switch problem. The residual norm reduction of 10^{-8} is achieved by FGMRESD with a Krylov subspace size of 100. The group-independent set size is 500 and the number of levels is 2. For all the levels of recursion, the fill-in parameter is 20 and the dropping tolerance is 10^{-4}.

3.4 Steady State Solution

This section compares several general methods derived to compute the steady state solution to the chemical master equation.

The first method is derived by extending the final time t_f large enough in the transient analysis. All three categories of methods discussed in Section 3 could be used under this setting. However, for extremely stiff problems, even the A-stable trapezoidal method may not be accurate unless the step size is small enough, as shown by the corresponding entries in Table 6 and Figure 12. This observation has also been made by Reibman and Trivedi [72].

Consider the equation $P_r^{(i)}(I - \frac{1}{\lambda}Q) = P_r^{(i-1)}$ from the implicit Euler algorithm. The system reaches steady state when $P_r^{(i)} = P_r^{(i-1)} = P_s$, which implies that P_s is in fact a left eigenvector of $(I - \frac{1}{\lambda}Q)$ corresponding to the eigenvalue one. In fact

$$P_s(I - \frac{1}{\lambda}Q) = P_s \Rightarrow P_sQ = 0. \tag{3.7}$$

Moreover, from the Perron-Frobenius Theorem [3] for nonnegative matrices, it can be shown that P_s is the only eigenvector corresponding to the eigenvalue one, and furthermore P_s is nonnegative. Consider the matrix $Q + \eta I$ where $\eta = \max_{1 \le i \le N} q_i$ as defined. The matrix $Q + \eta I$ is nonnegative and irreducible if all the reactions are coupled. By the

Gerschgorin circle theorem and the special structure of Q, the spectral radius of $Q + \eta I$ is η. Therefore, the Perron-Frobenius Theorem implies that η is a simple eigenvalue, which implies that 1 is a simple eigenvalue of the matrix $I - Q/\lambda = (1 + \eta/\lambda)I - 1/\lambda(Q + \eta I)$. Two types of methods are tried here to compute the particular eigenvector P_s. One is the Arnoldi method, which is implemented in the software package ARPACK [51]. The other one is the power method with an appropriate shift. The power method does not work well in practice, because while the dominant (real) eigenvalue η corresponds to P_s, there are other complex eigenvalues ζ with $|\zeta| = \eta$, and the power method cannot easily extract P_s from the subspace spanned by P_s and the other eigenvectors corresponding to the eigenvalues ζ.

The last and the most straightforward approach to compute the steady state solution is simply solving the linear system $P_s Q = 0$ combined with the constraint that $\|P_s\|_1 = 1$. Note that this (singular) linear system is extremely ill conditioned for iterative linear system solvers. However, this approach proved to be the most efficient one when the linear system is solved by direct sparse linear system solvers, which also require more computer memory than most other methods. See Table 6.

Method	Parameter	CPU time	Error
Implicit Euler	Direct	395.5	1.9×10^{-16}
Implicit Euler	Iterative	355.4	3.2×10^{-9}
Trapezoidal	Direct	326.3	2.6×10^{-8}
Trapezoidal	Iterative	657.4	2.7×10^{-8}
Krylov	$m = 60$	2156.9	3.9×10^{-11}
Arnoldi	$\lambda = 0.1$	278.2	8.5×10^{-8}
Arnoldi	$\lambda = 1$	348.5	1.2×10^{-9}
Power*	$\lambda = 1$	10593.1	1.7×10^{-6}
Direct		10.9	8.5×10^{-17}
Iterative		340.1	5.4×10^{-12}

* Aitken's δ^2 acceleration is applied when appropriate.

Table 6. CPU time (sec) and computational error for different algorithms. The final time t_f for transient methods is 2500 min. The error equals $\|\tilde{P}_s Q\|_\infty$, where $\tilde{P}_s$ is the numerical solution computed.

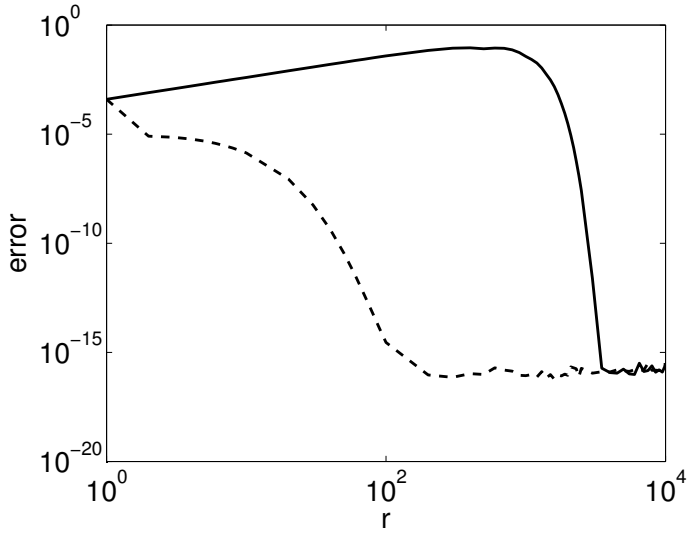


Figure 12. Computational error when using the ODE algorithms to compute the steady state solution. $t_f = 2500\,\text{min}$ and t_f/r is the time step size. Implicit Euler method, dashed line. Trapezoidal method, solid line.

3.5 Conclusions

This chapter mainly considers direct solutions to the chemical master equation using, basically, conventional numerical ODE solution techniques and recent Krylov space methods. For the ODE methods, due to the stiffness of the problem, only the implicit Euler method and the trapezoidal method are considered here. The numerical results indicate that the trapezoidal method has better accuracy but more computational cost than the implicit Euler method at the same step size, which matches the theoretical results. Krylov space methods are not quite as efficient as these two methods when the problem size is large.

Comparing to Monte-Carlo algorithms, the ODE solvers perform better than SSA, a Monte-Carlo algorithm, when the problem is under a certain size scale. One way to solve even larger models more effectively is to utilize parallel computation. This chapter shows some preliminary computation results on a parallel cluster using up to 16 processors. Parallel efficiency results on parallel clusters with more processors need to be verified. Another approach is to shrink the problem size by approximation techniques, which could be done on the modeling, process [6, 40], or numerical levels [58], which is the topic for the next two chapters.

Chapter 4: ADAPTIVE AGGREGATION METHOD FOR THE CME

4.1 Introduction

Suppose a chemical system consists of D different species and M reaction channels. If the size of the state space of each species S_i is N_i, then the total state space size $N = N_1 \times N_2 \times \cdots \times N_D$. As the number of species increases, the total state space size increases exponentially, known as the curse of dimensionality. Let $p(x,t)$ denote the probability mass of the state x at time t, where $x = (x_1, x_2, \ldots, x_D) \in \mathbf{Z}_{N_1} \times \mathbf{Z}_{N_2} \times \cdots \times \mathbf{Z}_{N_D}$ is a vector of integers representing one possible state of the system. It is worth noticing that there are cases where the number of molecules of a certain species S_i does not lie in $[0, N_i - 1]$ (i.e., the integers 0, 1, ..., $N_i - 1$ represent an encoding of the actual number of molecules) and/or the whole state space is not rectangular or even finite. However, most of the analysis presented in this chapter is also valid in those problem settings.

The chemical master equation (CME), derived from the Markov property of the underlying stochastic process [90, 39, 91, 72], is the ODE system describing the time evolution of the function $p(x,t)$ for every possible state x,

$$\frac{dP(X,t)}{dt} = P(X,t)Q,$$

where $P(X,t) = \big(p(x^{(1)},t),\, p(x^{(2)},t),\, \ldots\big)$ is the complete probability vector at time t and the vector $X = \big(x^{(1)},\, x^{(2)},\, \ldots\big)$ is a particular enumeration of the state space. Here Q is a constant sparse square matrix (called the infinitesimal generator of the system) with each of its components q_{ij} $(i \neq j)$ denoting the instantaneous rate at which the system makes one transition from one state $x^{(i)}$ to another state $x^{(j)}$ through one of the M prescribed reaction channels. Each row of the matrix Q sums up to zero, i.e., $q_{ii} = -\sum_{j \neq i} q_{ij}$, so that the whole system satisfies the conservation law of the overall probability mass. Now, each row of Q has at most $M + 1$ nonzero entries, hence the matrix Q is extremely sparse.

Various approximation methods have been proposed to reduce the size of the matrix Q. Consider aggregation/disaggregation operators E and F. The aggregation operator E maps any complete probability vector $P(X,t)$ to an aggregated probability vector $\tilde{P}(Y,t)$

and the disaggregation operator F does the opposite. In the discrete case, the operators E and F are just matrices. Now, the original ODE system can be condensed into a much smaller ODE system corresponding to the aggregated state space Y,

$$\frac{d\tilde{P}(Y,t)}{dt} = \tilde{P}(Y,t)FQE.$$

Note that this is essentially the model order reduction problem of control theory or mechanics. There are many different ways to choose appropriate E and F, determined by the measure of the distance from $P(X,t)$ to $\tilde{P}(Y,t)$. One idea, called the finite state projection algorithm [61, 9, 65], is to choose E and F such that FQE is a submatrix of the original matrix Q. Let $J \subset \{1, 2, \ldots, N\}$, and denote by X_J the subvector of X formed from the indices in J, and by Q_{JJ} the submatrix of Q with rows and columns indexed by J. Let $Y = (y^{(1)}, y^{(2)}, \ldots) = X_J$ denote the finite vector of states of specific interest, and the matrix Q_{JJ} be the submatrix of the matrix Q corresponding to the vector Y. Then with $\tilde{Q} = Q_{JJ}$, $\tilde{P} = P_J$, and $Y = X_J$, the condensed system to be solved is

$$\frac{d\tilde{P}(Y,t)}{dt} = \tilde{P}(Y,t)\tilde{Q}.$$

Since $\tilde{Q}$ is only part of the matrix Q, the overall probability mass for the new system no longer satisfies the conservation law. From a simulation point of view, in this new system any trajectory that reaches outside of the states Y before time t is lost forever. That is the major drawback of the finite state projection method. However, if the size of the original state space is infinite, this is the only way to reduce the original problem to a finite state problem.

Another idea is to first divide the state space into bins using grids, then let the aggregation operator E map states in the same bin into a single state in the reduced system [40, 41]. The probability mass of each single (reduced) state thus equals the sum of the probability masses of all those states mapped into it. One easy way to choose the corresponding disaggregation matrix F is to divide the probability mass of each single (reduced) state evenly into parts and assign this value as the probability mass of every state in the same bin that maps to that single (reduced) state by operator E.

In most chemical systems, it often happens that the probability masses of nearby states are very close to each other, therefore it is reasonable to combine these nearby states together to reduce the size of the problem. A more plausible way would be to apply the above reduction only to the part of the state space where probability mass is low and remains almost constant over time, not to the part of the state space where probability mass is high and changes significantly. In practice, collections of simulation results may be used to determine how to choose aggregation operators this way, leading to the adaptive aggregation method proposed here.

The remaining sections of this chapter are organized as follows. Section 4.2 demonstrates an adaptive way to implement aggregation methods, i.e., the adaptive aggregation method for the CME. Numerical results using this new approach are discussed in Section 4.3 and Section 4.4 concludes.

4.2 Adaptive Aggregation Method

In aggregation methods, aggregated grids and their aggregation/disaggregation operators E and F may be determined statically or dynamically. In the static case, the grids are determined at the beginning of the computation process and never changed after that, while in the dynamic case the grids adjust to the dynamics of the computation. Dynamic gridding makes more sense when the computation domain that matters most is much smaller than the whole state space and changes over time.

When solving the CME, dynamic gridding for the aggregation technique means coarse grids for the states with low probabilities and fine grids for the states with high probabilities. One simple way to distinguish these two groups of states is by Monte Carlo methods such as SSA. The whole state space is first divided into a suitable number of bins. Let $t_0 = 0$ be the initial time, $t_n = t_f$ the final time, and check times t_0, t_1, t_2, ..., t_n equally spaced in the time interval $[t_0, t_f]$. Simulate the dynamics of the chemical system from t_0 through t_n several times using Monte Carlo methods like SSA, and on each time interval $[t_i, t_{i+1}]$ $(0 \leq i \leq n-1)$ record all the bins that have been touched by at least one trajectory between time t_i and t_{i+1}. Now, all the states within these recorded bins are categorized as states with

likely high probability masses, whereas all other states are categorized as states with likely low probability masses. When integrating the original ODE system from time t_i through time t_{i+1}, choose the aggregation operator E such that all the states that have been marked as states with likely high probability masses remain the same after the mapping, and all other states, namely states with likely low probability masses, are aggregated into reduced states according to the aggregation grids.

Here are two numerical examples on a simple birth-death process given by

$$\left.\begin{array}{lll} \emptyset & \stackrel{k}{\rightarrow} & x \\ x & \stackrel{\mu}{\rightarrow} & \emptyset \end{array}\right\},$$

where $k = 1.0s^{-1}$, $\mu = 0.01s^{-1}$. Figure 13 shows the numerical result when the reduced model is constructed using information from 10 SSA simulations, all started from the same initial state, $x_1(0) = 10$. The aggregation matrix is then formed based on these 10 trajectories shown in Figure 13(A). For example, on the time (seconds) interval $[60, 80]$, every state in the closed interval $[49, 80]$ is not aggregated, while all other states are aggregated according to the grids on the state space. Hence, the size of the state space after aggregation is $4 \times 8 + (16 - 4) = 44$, much less than the size of the original state space, $16 \times 8 = 128$. Figure 14 shows the fact that increasing the number of SSA runs can improve the approximation accuracy of the reduced model at the expense of additional computational cost. For instance, in this case with 20 SSA simulations every state in the closed interval $[33, 80]$ is not aggregated on the same time interval $[60, 80]$. Thus, the number of states after aggregation is now $6 \times 8 + (16 - 6) = 58$, larger than the previous case as expected, however, the numerical solutions in Figure 14(B) are smoother and closer to the actual solutions than in Figure 13(B).

The only constraint for the corresponding disaggregation matrices F is that the aggregation/disaggregation matrix pair E and F should satisfy the constraint that $FE = I$,

where I is a square identity matrix, like

$$FE = \begin{pmatrix} \frac{1}{k} & \frac{1}{k} & \cdots & \frac{1}{k} & & & & & & & & \\ & & & & 1 & & & & & & & \\ & & & & & 1 & & & & & & \\ & & & & & & \ddots & & & & & \\ & & & & & & & 1 & & & & \\ & & & & & & & & \frac{1}{k} & \frac{1}{k} & \cdots & \frac{1}{k} \end{pmatrix} \begin{pmatrix} 1 & & & & & & & & & & & \\ 1 & & & & & & & & & & & \\ \vdots & & & & & & & & & & & \\ 1 & & & & & & & & & & & \\ & 1 & & & & & & & & & & \\ & & 1 & & & & & & & & & \\ & & & \ddots & & & & & & & & \\ & & & & 1 & & & & & & & \\ & & & & & & & & 1 & & & \\ & & & & & & & & 1 & & & \\ & & & & & & & & \vdots & & & \\ & & & & & & & & 1 & & & \end{pmatrix} = I,$$

where the integer k is the bin size. Take $P = (p_1, p_2, \ldots, p_{3k})$ as any complete probability vector. Then $PE = (\sum_{i=1}^{k} p_i,\ p_{k+1},\ p_{k+2},\ \ldots,\ p_{2k},\ \sum_{i=2k+1}^{3k} p_i) = \tilde{P} = (\tilde{p}_1,\ \tilde{p}_2,\ \ldots,\ \tilde{p}_{k+2})$ produces the reduced probability vector $\tilde{P}$ of vector P, and $\tilde{P}F = (\frac{1}{k}\tilde{p}_1,\ \ldots,\ \frac{1}{k}\tilde{p}_1,\ \tilde{p}_2,\ \ldots,\ \tilde{p}_{k+1},\ \frac{1}{k}\tilde{p}_{k+2},\ \ldots,\ \frac{1}{k}\tilde{p}_{k+2})$ maps the reduced probability vector $\tilde{P}$ back to the complete probability vector space by dividing $\tilde{p}_1$ and $\tilde{p}_{k+2}$ into k parts.

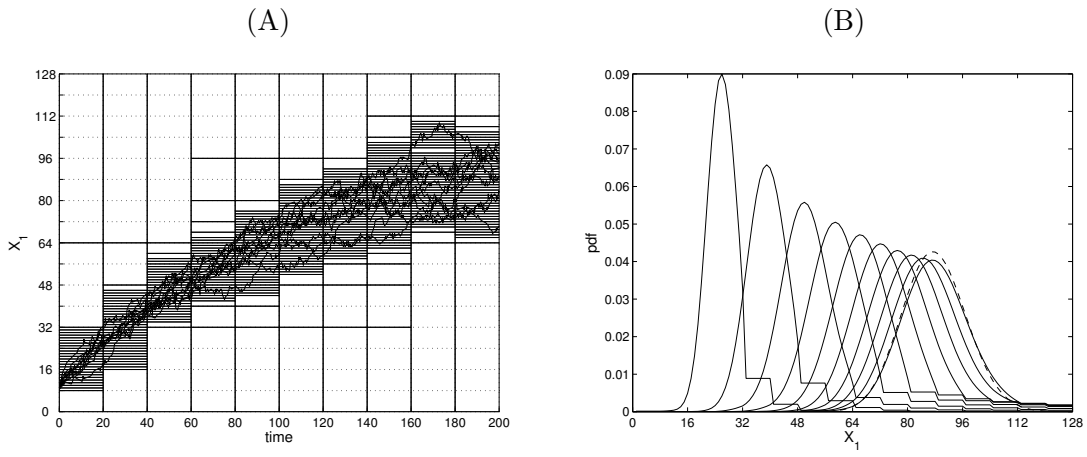


Figure 13. (A) Ten SSA simulations of the simple birth-death process with $t_f = 200$. The grids on the graph divide the state space into bins and the time space into subintervals. (B) Computational results from the full model and the reduced model. The sequence of solid lines shows the solutions to the reduced model at each time $t_i = 20i$, $1 \le i \le 10$. The dashed line, which is very close to the solid line at the time t_{10}, is the solution to the full model at the final time $t_f = t_{10}$.

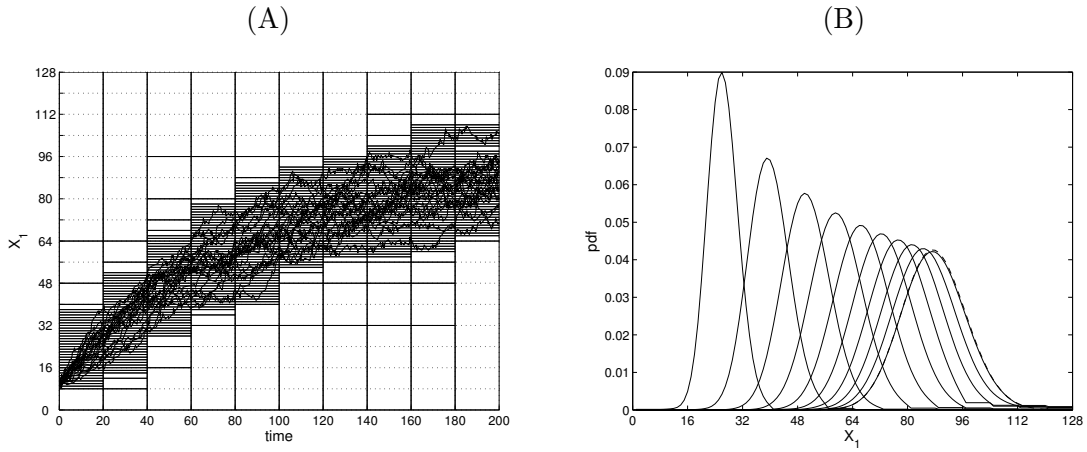
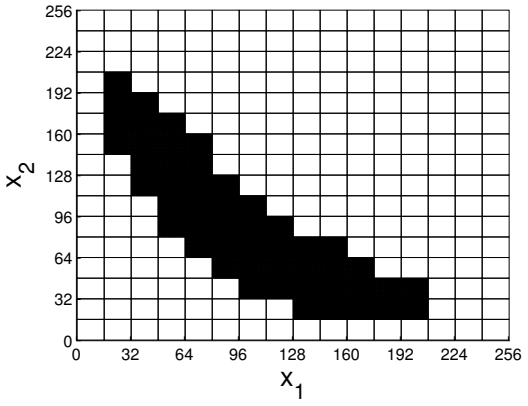


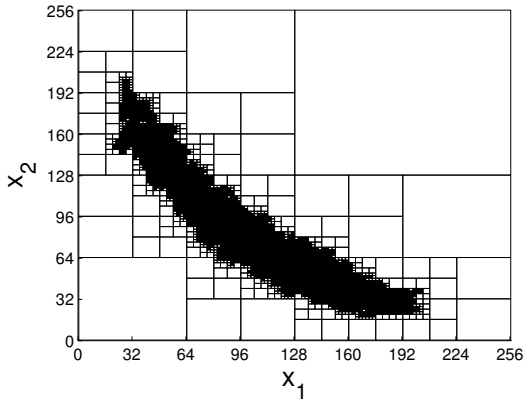
Figure 14. (A) Twenty SSA simulations of the simple birth-death process with $t_f = 200$. (B) Computational results from the full model and the reduced model. The sequence of solid lines shows the solutions to the reduced model at each time $t_i = 20i$, $1 \leq i \leq 10$. The dashed line, which almost coincides with the solid line at the time t_{10}, is the solution to the full model at the final time $t_f = t_{10}$.

Assume that the state space is divided evenly into M bins and the size of each bin is k, that is, $N = Mk$. Furthermore, assume the number of bins that has been reached by all the trajectories on each time interval is relatively small compared to M. Then the size of the reduced model is $O(k) + O(M)$, which reaches its minimum when M and k are approximately balanced, $M = \Theta(k)$, and so the minimum obtained is $O(\sqrt{N})$. This implies that the size of the state space would be reduced to its square root at best, and thus unequal size bins are required for any further reduction. One efficient way to do that is to use bins with different sizes in a hierarchic way as shown by the solid straight lines in Figures 13, 14, and in Figure 15(B) as a two-dimensional example. Starting from the whole computation domain, each bin is recursively partitioned evenly into two/four smaller bins unless it has never been touched by any trajectory during the time interval. In this way the size of the reduced model is further reduced. For comparison, the number of states in the reduced model using the grid in Figure 15(A) is 13006, while the number of states using the grid in Figure 15(B) is 8172, a reduction of approximately 37%. The recursive partitioning is even better in higher dimensions, e.g., in d dimensions, large blocks of I^d states are reduced to a single state.

(A) Gridding with fixed bin size



(B) Gridding with adaptive bin size



(C) States reached by SSA runs

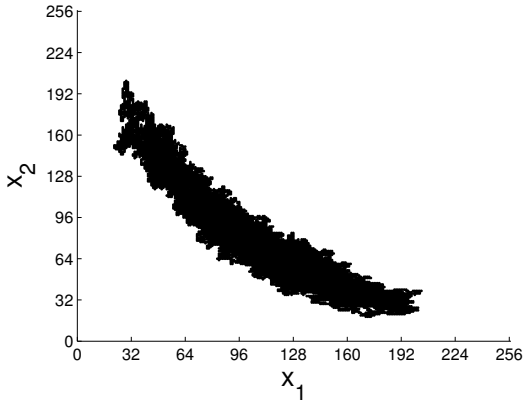


Figure 15. Comparison of two different griddings for the adaptive aggregation method on the toggle model.

4.3 Numerical Results

This section presents some more numerical experiments on molecular biology models.

4.3.1 Schlögl Reaction

One potential problem with the finite state projection algorithm is its inefficiency when dealing with bistable distributions. When the probability distribution is bistable, the best reduced model should only focus on the two stable states. However, if all the states between these two stable states are wiped out completely as could happen with the finite state projection algorithm, then the innate bistable nature of the model is lost. In the proposed adaptive aggregation method, however, this will not happen, since all the states with small

probabilities are not simply thrown away but aggregated according to the coarse grids so that the bistable property of the original system is preserved. This speculation can be verified from numerical experiments on the well-known bistable Schögl reaction (34), which is given by

$$\left.\begin{array}{rcl} b_1 + 2x & \underset{c_2}{\overset{c_1}{\rightleftharpoons}} & 3x \\ b_2 & \underset{c_4}{\overset{c_3}{\rightleftharpoons}} & x \end{array}\right\},$$

where b_1 and b_2 are buffered species whose populations are assumed to be constant and the parameters here are $c_1 = 3 \times 10^{-7} s^{-1}$, $c_2 = 10^{-4} s^{-1}$, $c_3 = 10^{-3} s^{-1}$, $c_4 = 3.5 s^{-1}$, $b_1 = 1 \times 10^5$, $b_2 = 2 \times 10^5$, and the final time $t_f = 10.0s$. The Schlögl reaction does not model any known chemical system, since there are no actual trimolecular reactions in nature, however, it is often used in numerical demonstrations because of its bistable property.

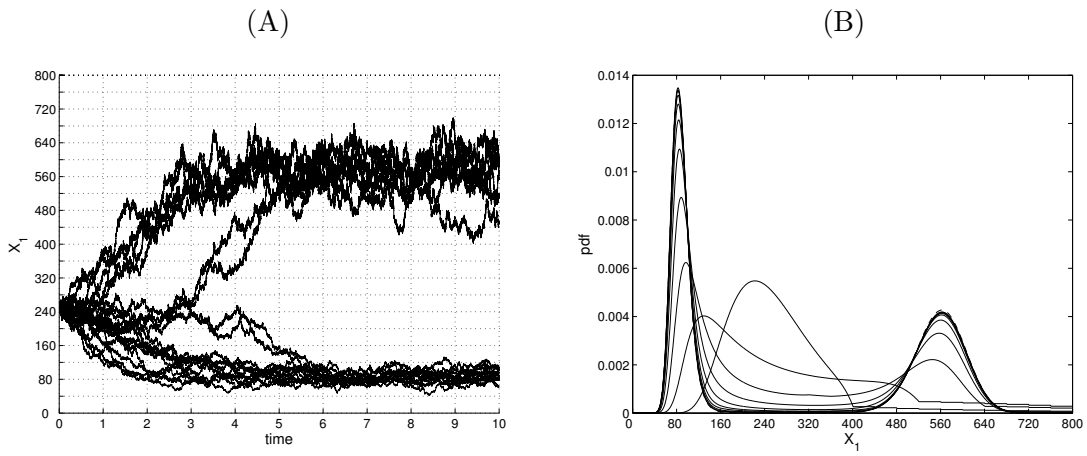


Figure 16. (A) Twenty SSA simulations of the Schlögl reaction with $t_f = 10.0$. The grids on the graph divide the state space into bins and the time space into subintervals. (B) Computation results from the full model and the reduced model. The sequence of solid lines are the solutions to the reduce model at each time $t_i = i$, $1 \leq i \leq 10$. The dashed line, which coincides with the solid line at the time t_{10}, is the solution to the full model at the final time $t_f = t_{10}$.

Figure 16(A), comprised of twenty trajectories drawn from SSA simulations, shows qualitatively the bistable property of the Schlögl reaction starting from the initial state $x_1(0) = 250$. This bistable behavior is presented quantitatively in Figure 16(B), which plots the results from the reduced model and full model. Figure 16(B) also shows that the final solution from the reduced model matches well with the solution from the full model, however, the reduced model requires only half the computational effort.

4.3.2 Toggle Switch Model

The toggle switch model, which has often been used to model synthetic bistable gene regulatory networks [30], is another molecular biology model with bistable behavior. One such model can be characterized by the following reaction equations [21],

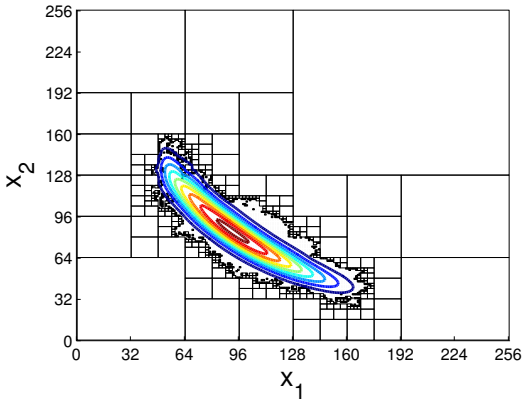
$$\left.\begin{array}{lll} \emptyset & \xrightarrow{\alpha/(\beta+x_2^2)} & x_1 \\ x_1 & \xrightarrow{\mu} & \emptyset \\ \emptyset & \xrightarrow{\gamma/(\delta+x_1^2)} & x_2 \\ x_2 & \xrightarrow{\mu} & \emptyset \end{array}\right\}$$

with parameters $\alpha = \gamma = 1200\,\mathrm{min}^{-1}$, $\beta = \delta = 6000$, and $\mu = 10^{-3}\,\mathrm{min}^{-1}$. Here, X_1 and X_2 can be seen as two competing proteins, and the toggle comprises two switches, where each protein represses the production of the other.

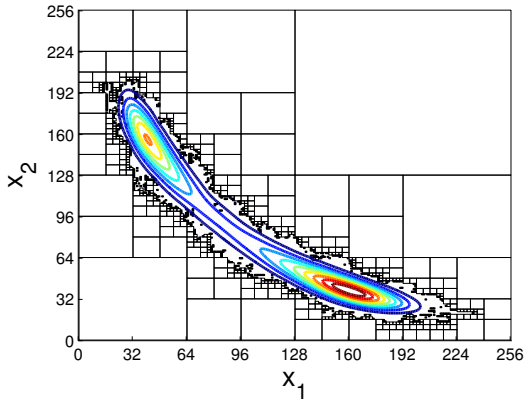
Figure 17 demonstrates how the proposed algorithm works adaptively on this specific problem starting from $x_1(0) = 110$, $x_2(0) = 90$, on the time (minutes) interval $[t_0, t_f] = [0, 50000]$. The whole time interval is divided evenly into ten subintervals by grid points $t_i = 5000i$, $0 \leq i \leq 10$. Figure 17 shows the computation result (contour plots of the aggregated probability vector $\tilde{P}(Y, t)$) together with the griddings. Altogether, these three graphs demonstrate how the aggregation algorithm is applied adaptively and efficiently for the toggle switch model.

Figure 18 compares the estimated marginal probability distributions of species X_1 and X_2 from three different methods. The computational times are listed in Table 7, with the SSA algorithm implemented in Stochkit [52] and the CMEs solved by the Krylov space projection method, implemented in Expokit [88]. The runtime reduction for the reduced model is not as great as might have been expected since the size of the reduced model is still comparable to the full model and the proposed algorithm also introduces overhead computing the matrices E and F and matrix vector multiplications using E and F. However, the reduction in computational effort will become more significant if the problem size is large enough, as shown in the next example.

(A) After 5000 min



(B) After 20000 min



(C) After 50000 min

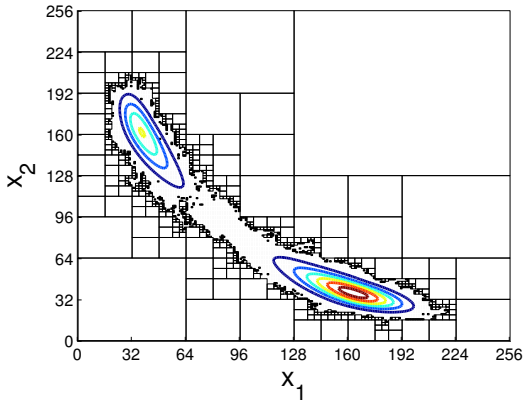


Figure 17. The evolution of the adaptive gridding and the numerical solution $\tilde{P}(Y,t)$ to the reduced model based on the gridding for the toggle switch problem.

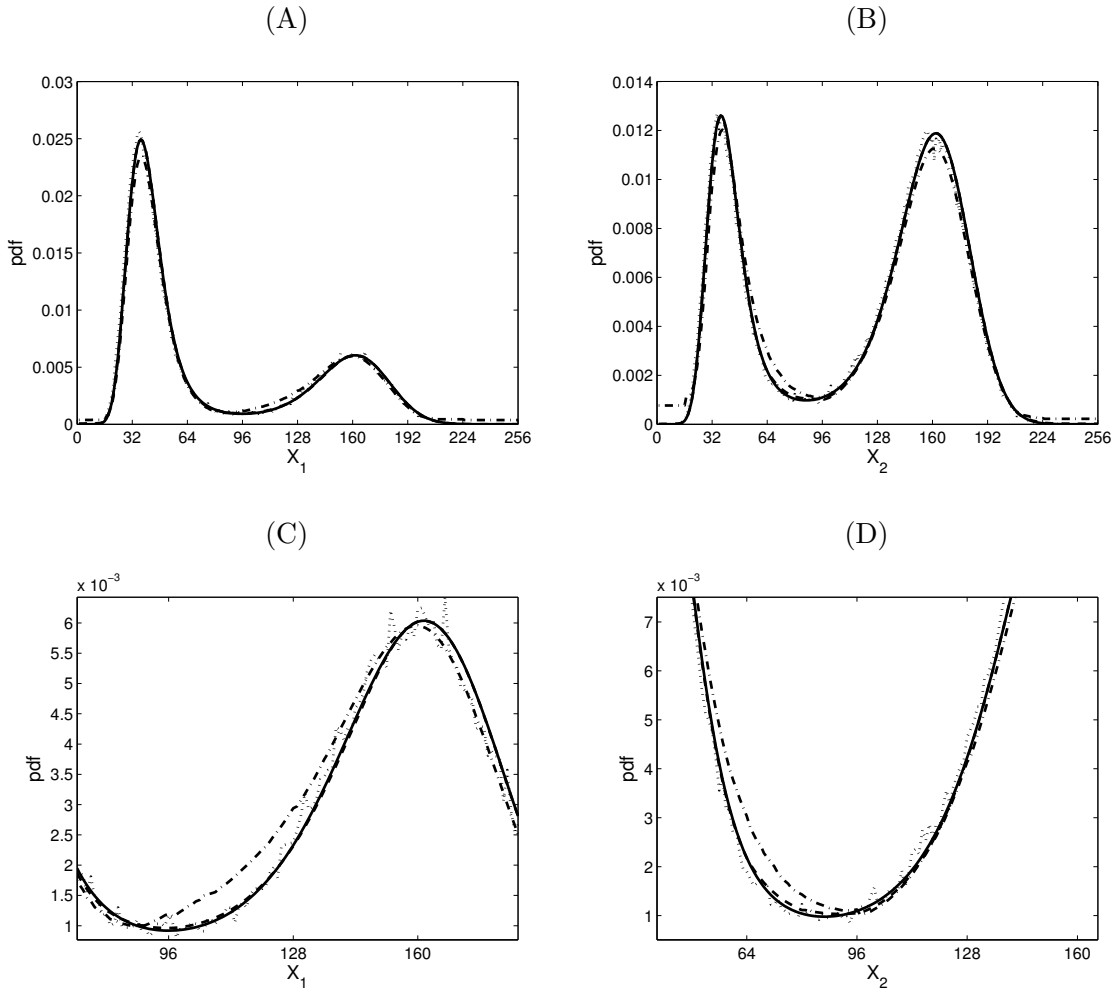


Figure 18. Numerical estimates for marginal probability distributions using the full model (solid lines), the reduced model with fixed bin size 16×16 (dashed lines), the reduced model with adaptive bin size (dash-dot lines), and histograms from 10^5 SSA simulation runs (dotted lines): (a) marginal probability distribution of X_1, (b) marginal probability distribution of X_2, (c) close-up of (a), (d) close-up of (b).

4.3.3 A Simple Cell Cycle Model

This section focuses on a simple cell cycle model derived from some normalized phenomenological rate equations [94]. The model can be described by the following elementary reactions without intermediates and with variable propensity rates:

$$\left.\begin{array}{rcl}
\emptyset & \underset{k_2'}{\overset{k_1 m V_s}{\rightleftharpoons}} & x_{CycB} \\
x_{CycB} + x_{Cdh1} & \overset{k_2''/V_s}{\longrightarrow} & x_{Cdh1} \\
x_{CycB} + x_{Cdh1} & \overset{\frac{k_4}{J_4 V_s + x_{Cdh1}}}{\longrightarrow} & x_{CycB} + x_{Cdh1P} \\
x_{Cdc20} + x_{Cdh1P} & \overset{\frac{k_3''}{J_3 V_s + x_{Cdh1P}}}{\longrightarrow} & x_{Cdc20} + x_{Cdh1} \\
\emptyset & \underset{k_6}{\overset{k_5' V_s + k_5'' V_s \frac{x_{CycB}^2}{(J_5 V_s)^2 + x_{CycB}^2}}{\rightleftharpoons}} & x_{Cdc20}
\end{array}\right\}.$$

Here, parameters $k_1/c_{CycB} = 0.01\,\mathrm{min}^{-1}$, $k_2' = 0.04\,\mathrm{min}^{-1}$, $k_2'' c_{Cdh1} = 1.0\,\mathrm{min}^{-1}$, $k_3'' c_{Cdc20}/c_{Cdh1} = 10.0\,\mathrm{min}^{-1}$, $k_4 c_{CycB}/c_{Cdh1} = 35\,\mathrm{min}^{-1}$, $k_5'/c_{Cdc20} = 0.005\,\mathrm{min}^{-1}$, $k_5''/c_{Cdc20} = 0.2\,\mathrm{min}^{-1}$, $k_6 = 0.1\,\mathrm{min}^{-1}$, $J_3/c_{Cdh1} = 0.04$, $J_4/c_{Cdh1} = 0.04$, $J_5/c_{CycB} = 0.3$, where c_S is the characteristic concentration of the species S. The cell mass variable m is introduced to reflect the assumption that the species $CycB$ is synthesized at a supralinear rate such that its molecular number increases with cell mass and V_s, the nominal volume of the cell times Avogadro's number, is chosen to be 18 molecules/nMolar. The species $Cdh1P$ is the phosphorylated form of $Cdh1$ so that $x_{Cdh1} + x_{Cdh1P} = c_{Cdh1} V_s$.

Numerical experiments are conducted here under the assumption of characteristic concentrations, $c_{CycB} = c_{Cdh1} = c_{Cdc20} = 3.5nM$, and the final time (minutes) $t_f = 70.0$. Figure 19 compares the numerical results from different numerical schemes and Table 7 lists their CPU times (in seconds).

Table 7 shows that the CPU times of the proposed algorithm are comparable to those of the Monte Carlo method SSA and much less than solving the CME on the full state space. Actually, the reduction in computational effort will become even more substantial as the problem size grows.

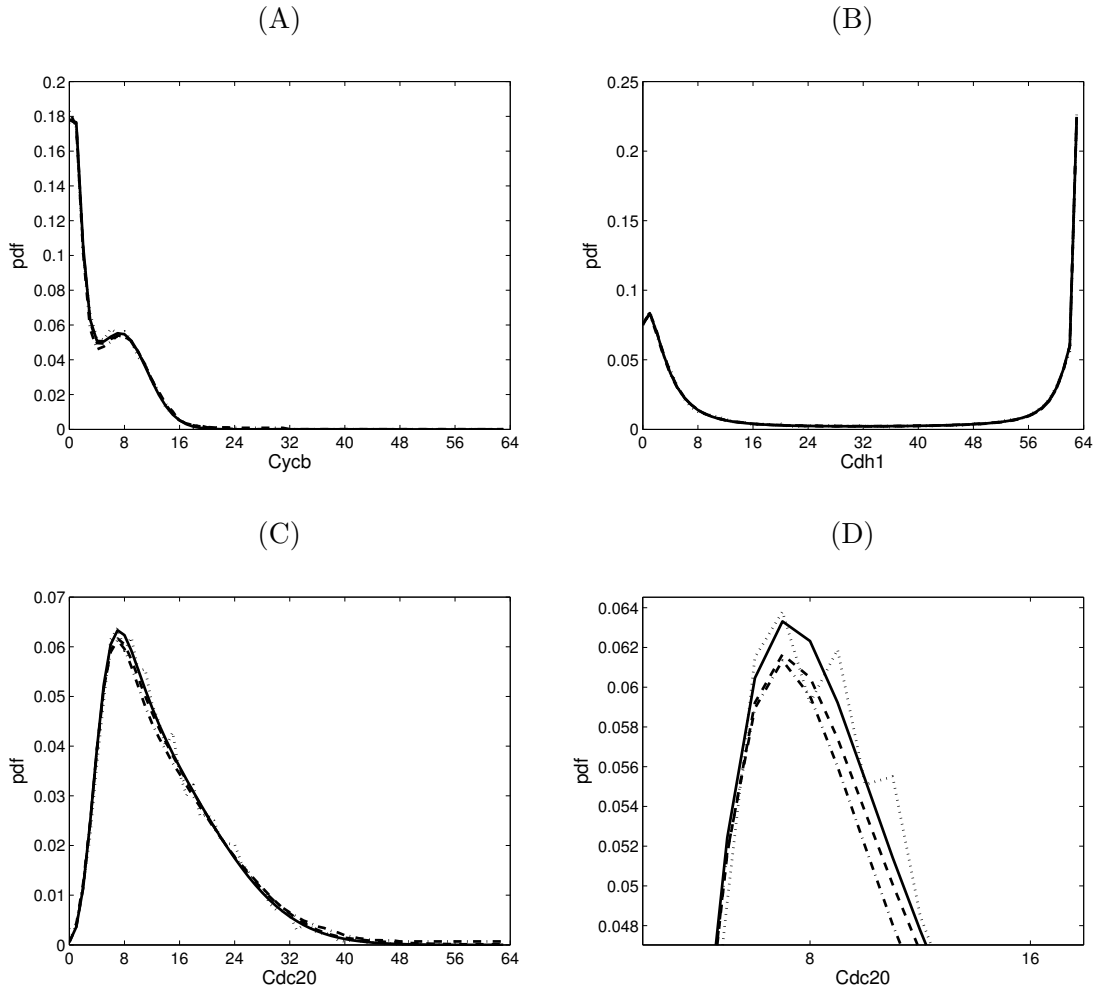


Figure 19. Numerical estimates for marginal probability distributions at time (minutes) $t_f = 70.0$, $c_{CycB} = c_{Cdh1} = c_{Cdc20} = 3.5nM$, the full CME model (solid lines), the reduced CME model with fixed bin size $8 \times 8 \times 8$ (dashed lines), the reduced CME model with adaptive bin size (dash-dot lines), and the histogram from 10^4 SSA simulation runs (dotted lines): (a) marginal probability distribution of $Cycb$, (b) marginal probability distribution of $Cdh1$, (c) marginal probability distribution of $Cdc20$, (d) close-up of (c). Note that the CME curves are much smoother than the SSA histogram. Note that the CME curves are much smoother than the SSA histogram.

4.4 Conclusions

One major challenge when using the chemical master equation to model gene regulatory networks and some other biological systems is that it typically requires integrating some very large ODE systems to get solutions to the CME [59]. This chapter contributes a way to use information from approaches like Monte Carlo methods to decrease the size of the CME ODE system, as shown in the numerical examples. The idea is to combine "dimension-free"

CPU time (in seconds) for different methods.

model	Toggle switch	Cell cycle
SSA[1]	1.6×10^3	125.2
CME(full)	78.3	4.7×10^3
CME(reduced)[2]	36.5	353.0
CME(reduced)[3]	18.8	137.5

Average sizes of the linear systems in the CME for different methods.

model	Toggle switch	Cell cycle
CME(full)	6.6×10^4	2.6×10^5
CME(reduced)[2]	1.4×10^4	3.4×10^4
CME(reduced)[3]	8.6×10^3	1.6×10^4

1. 10^5 SSA runs for the toggle switch model and 10^4 runs for the cell cycle model. (The toggle switch model needs more SSA runs because of its bistable property.)
2. Reduced model with fixed bin sizes.
3. Reduced model with adaptive bin sizes.

Table 7. Computational effort for different methods.

Monte Carlo methods with "curse of dimensionality"-suffering deterministic algorithms. The decrease in computational effort is more significant as the dimension of the problem grows, but then the ODE system size also grows exponentially with the dimension.

One potential problem with the proposed algorithm is the difficulty estimating the model reduction error. Empirically, this error usually depends on the number of simulation runs used to construct the reduced model, as shown in the simple birth-death example in Section 4.2. Usually, increasing the number of simulation runs expands the part of state space marked as states with likely high probability mass, thus reducing the approximation error. For a similar reason, the proposed algorithm works best when the statistical variance of the system is relatively small and not so well if the variance is high, which often occurs when the chemical system is bistable. Nevertheless, for the numerical examples presented here, where the variance is relatively high, the performance of the proposed algorithm is satisfactory.

Information collected from Monte Carlo methods is used here in a quite simple way in the proposed algorithm. A more sophisticated approach would be to first construct a surrogate model [13] (mathematical approximation) using data from Monte Carlo methods,

and then reduce the size of the ODE system based on the surrogate model, instead of using data from Monte Carlo simulations directly. The idea would be to adapt grids according to the smooth estimated probability mass function, which is likely more accurate than simple histograms.

Chapter 5: RADIAL BASIS FUNCTION COLLOCATION FOR THE CME

5.1 Introduction

In a well-mixed, fixed-temperature, and fixed-volume chemical system of D reacting species, the chemical master equation gives an accurate description of the probability $p(x,t)$ that the chemical system will have a molecular population vector $x \in \mathbf{N}^D$ at time t, where the i-th component of x represents the molecular number of species i [95, 33]:

$$\frac{\partial p(x,t)}{\partial t} = -p(x,t)\sum_{\mu=1}^{M} a_\mu(x) + \sum_{\mu=1}^{M} p(x-\nu_\mu,t)a_\mu(x-\nu_\mu).$$

$a_\mu(x)$ and $\nu_\mu \in \mathbf{Z}^D$ are the nonnegative propensity function and the stoichiometric transition vector, respectively, for reaction channel $\mu \in \{1, 2, \ldots, M\}$. When the state space $\{x^1, x^2, \ldots\}$ (each element x^i is a distinct D-dimensional molecular population vector) is chosen, the chemical master equation can be rewritten into an infinite linear system of ODEs (ordinary differential equations),

$$\frac{dP(t)}{dt} = P(t)A, \tag{5.1}$$

where $P(t)$ is the complete probability row vector at time t, $P(t) = (p(x^1,t), p(x^2,t), \ldots)$. The time-independent matrix A is defined from the nonnegative propensity functions,

$$A_{ij} = \begin{cases} -\sum_{\mu=1}^{M} a_\mu(x^i), & \text{for } i = j; \\ a_\mu(x^i), & \text{for } j \text{ such that } x^j = x^i + \nu_\mu; \\ 0, & \text{otherwise.} \end{cases}$$

It can be seen from the definition above that A is extremely sparse with at most $M+1$ nonzero elements in each row and each row of A sums up to zero.

Numerical approaches solving the CME directly as a linear differential equation can be roughly categorized into two groups, grid-based methods and Galerkin-based ones [47]. This chapter blends the ideas from these two groups of approaches into a new method. First, a grid-based method is used to divide the computation domain into a collection of bins. Then, on each of these bins, a Galerkin-based method is applied, where the solution $P(t)$ is projected onto a finite dimensional function space. Radial basis functions (RBF) are introduced to improve the approximation in high dimensions. Hopefully, this new method will produce a much larger reduction in the computational effort to solve the CME, while at the same time maintaining accuracy.

5.1.1 Radial Basis Functions

Consider a one-dimensional Schlögl model based on chemical reactions

$$\left.\begin{array}{rcl} b_1 + 2x & \underset{c_2}{\overset{c_1}{\rightleftharpoons}} & 3x \\ b_2 & \underset{c_4}{\overset{c_3}{\rightleftharpoons}} & x \end{array}\right\},$$

where b_1 and b_2 are buffered species whose respective molecular populations are constant, $b_1 = 10^5$, $b_2 = 2 \times 10^5$. The reaction rate constants are $c_1 = 3 \times 10^{-7}$, $c_2 = 10^{-4}$, $c_3 = 10^{-3}$, $c_4 = 3.5$ [34].

The Schlögl model shows the bistable property when the initial state is well-chosen and has been used as a good test problem for numerical methods. The chemical system has two stable states $x_{s_1} = 82$, $x_{s_2} = 563$ and one barrier state $x_b = 248$. The bistable property of the distribution function is apparent when the initial state is close to the barrier state, such as $x_0 = 240$ used here. The solid line in Figure 20(A) plots the distribution function $p(x, t)$ at $t = 5$.

Pictures like Figure 20(A) suggest that the exact solution $p(x, t)$ to the chemical master equation might be closely approximated by a linear combination of some well chosen basis functions,

$$p(x, t) \approx \sum_n b_n(t) F_n(x). \tag{5.2}$$

This chapter considers a special class of basis functions, namely radial basis functions $F_n(x)$ [68]. A real-valued function $\varphi(x)$ is *radial* if $\varphi(x) = \phi(\|x\|_2)$, where $\| \cdot \|_2$ is the Euclidean norm and $\phi(\cdot)$ is a real-valued function defined on $[0, \infty)$. Radial functions are usually used to construct function approximations as linear combinations of translates $\{\varphi(x - x^{s_1}), \varphi(x - x^{s_2}), \ldots, \varphi(x - x^{s_N})\}$, where $\{x^{s_1}, x^{s_2}, \ldots, x^{s_N}\}$ are distinct centers. Radial basis functions (RBF), such as the Gaussian radial basis function $\varphi(x) = e^{-\alpha \|x\|_2^2}$, are often used in nonparametric estimations of probability density functions. Another benefit is that RBFs are easy to implement and evaluate for a high dimensional state variable x, which is crucial for numerically solving CMEs containing more than three species.

Suppose that

$$p(x, t) = \sum_{n=1}^{N} b_n(t) \varphi(x - x^{s_n}),$$

and rewrite this into a matrix form for the vector $P(t)$, $P(t) = B(t)\Phi$, where $B(t) = (b_1(t), b_2(t), \ldots, b_N(t))$ and the matrix $(\Phi)_{ij} = \varphi(x^j - x^{s_i})$.

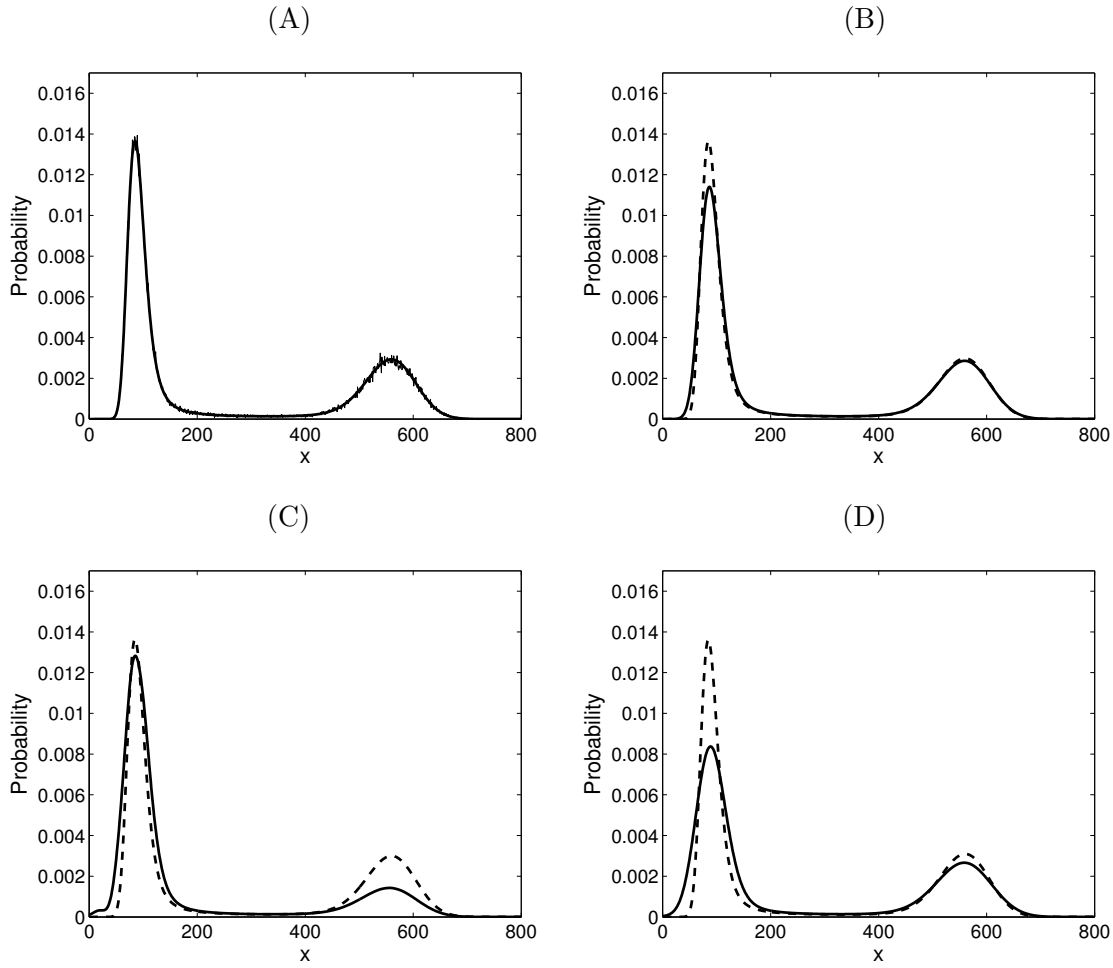


Figure 20. Numerical results for the Schlögl model.
(A) Thick solid line: CME solution with the state space restricted to [0 800]. Thin solid line: estimation from 100,000 SSA runs.
(B)–(D) Solid lines: collocation results using different Gaussian functions ϕ, (B): grid size 10, $\phi(r) = e^{-(r/15)^2}$; (C): grid size 20, $\phi(r) = e^{-(r/18)^2}$; (D): grid size 20, $\phi(r) = e^{-(r/30)^2}$. Dashed lines: CME solution with the state space restricted to [0 800].

5.1.2 Collocation Method

Substitute $P(t) = B(t)\Phi$ into the chemical master equation (5.1) giving

$$\frac{dB(t)}{dt}\Phi = B(t)\Phi A.$$

This is an overdetermined system for the vector $B(t)$, which means this equation could only be satisfied at a number of points in the state domain (called collocation points). Let $\{x^{s_1}, x^{s_2}, \ldots, x^{s_N}\}$, the set of centers for the radial basis functions, also be the set of collocation

points, and define a projection matrix F such that

$$P(t)F = \tilde{P}(t) = \big(p(x^{s_1}, t), \ldots, p(x^{s_N}, t)\big).$$

Then projecting (5.1) with F gives

$$\frac{dB(t)}{dt}\Phi F = B(t)\Phi AF, \tag{5.3}$$

where $(\Phi F)_{ij} = \varphi(x^{s_j} - x^{s_i})$ [60]. It is important that this matrix product ΦF be invertible, and it is computationally advantageous if the function $\phi(\cdot)$ has compact support (vanishes outside a bounded set), which implies the matrix ΦF is banded. Schoenberg [84] has proved that there is no compactly supported univariate function $\phi(\cdot)$ that generates nonsingular ΦF for any space dimension and any set of distinct center points. However, if $f(r)$ is nonconstant and completely monotone (e.g., $f(r) = e^{-r}$) with support $[0, \infty)$, and $\varphi(x) = f(\|x\|_2^2)$, then ΦF is positive definite for *any* dimension d and *any* N distinct center points. (f is *completely monotone* if $f \in C[0, \infty)$, $f \in C^\infty(0, \infty)$, and $(-1)^k f^{(k)}(r) \geq 0$ for $r > 0$ and all $k = 0$, 1, 2, ...)

Wendland [99] has proposed a class of piecewise polynomials as the function $\phi(\cdot)$ with compact support $[0, 1]$,

$$\begin{aligned}
\phi_{1,0}(r) &= (1-r)_+, \\
\phi_{1,1}(r) &= (1-r)_+^3(3r+1), \\
\phi_{1,2}(r) &= (1-r)_+^5(8r^2+5r+1), \\
\phi_{3,0}(r) &= (1-r)_+^2, \\
\phi_{3,1}(r) &= (1-r)_+^4(4r+1), \\
\phi_{3,2}(r) &= (1-r)_+^6(35r^2+18r+3)/3.
\end{aligned}$$

In general, $\phi_{d,k}(r) = I^k\big[(1-r)_+^{\lfloor d/2 \rfloor + k + 1}\big]$ scaled so that $\phi_{d,k}(0) = 1$, where $I[\phi](r) = \int_r^\infty t\phi(t)dt$ and $(t)_+ = \max\{0, t\}$. The function $\phi_{d,k} \in C^{2k}[0, \infty)$ induces RBFs $\varphi(x - x^{s_i}) = \phi(\|x - x^{s_i}\|_2)$ that would then generate a banded nonsingular Gramian matrix ΦF for any

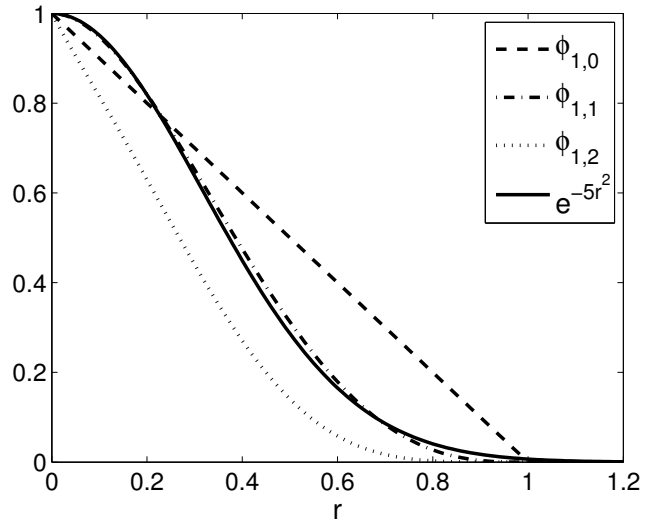


Figure 21. Plots of the Wendland functions $\phi_{1,0}(r)$, $\phi_{1,1}(r)$, $\phi_{1,2}(r)$, and the Gaussian function e^{-5r^2}.

set of distinct center points in d-dimensional space. This mathematical property also holds true for $\phi_{d,k}(r/c)$, which has compact support $[0, c]$ instead of $[0, 1]$.

As shown in Figure 21, some functions $\phi_{d,k}$ can be very close to the Gaussian function e^{-cr^2}. Even though the Gaussian function has infinite support, it can be closely approximated by assuming that its function value vanishes for large arguments (which is exactly true in floating point arithmetic). To guarantee that ΦF is invertible, theoretically either $\phi(r)$ has to have infinite support (from the Schönberg theory with $\varphi(x) = f(\|x\|_2^2)$) or any $\varphi(x) = \phi_d(\|x\|_2)$ having compact support must depend on the dimension d in a complicated way. The point is that computationally truncated Gaussians and the $\phi_{d,k}$ are very similar, hence truncating Gaussians to get a banded matrix ΦF is reasonable. In the numerical examples presented here, the Gaussians are truncated to zero when the function value is less than 10^{-8}.

5.1.3 Algorithm

First, the whole state space is divided into equally sized bins, such that all bins have the same edge length b_i in dimension i, since all radial basis functions are of the same shape. Therefore, the volume of each bin is $\prod_{i=1}^{D} b_i$. The bins are used for the placement of basis functions, but not all of them.

Then, run the SSA algorithm 20–500 times, depending on the size of the state space. Collect all the bins that have been touched by at least one simulation trajectory and denote the center points of all these bins as $\{x^{s_1}, x^{s_2}, \ldots, x^{s_N}\}$, which would then be used as the center points for the radial basis functions in the next step.

Next, select the function ϕ with appropriate parameters and build the implicit ordinary differential equation (5.3). Also solve the linear equation

$$B(0)\Phi F = \tilde{P}(0)$$

to get the initial condition $B(0)$ for the ODE system (5.3). Integrate (5.2) with the initial condition $B(0)$ until the final time t_f to get $B(t_f)$, giving $B(t_f)\Phi \approx P(t_f)$.

The most important step in the algorithm is the choice of ϕ and its shape parameters, which, in fact, is an important on-going research topic ([98] studied the relation between shape parameters and the condition number of the matrix ΦF, [83] and others discussed variable shape parameter strategies for RBF approximations). For example, in the Schlögl model mentioned above, numerical results in Figure 20 show the importance of the shape parameter c for the function $\phi(r) = e^{-(r/c)^2}$. (For grid size 10 (20), 20 SSA runs were done, and $N = 79$ (39), all starting from $x(0) = 240$.)

The optimal value of c seems dependent on the bin size and the shape of the probability distribution function $p(x, t)$ around that bin in this case. Using the same set of center points, a smaller c usually makes it easier to capture the narrow peaks, while a larger value of c means the approximation of the wider peaks are likely to be more accurate. More empirical results on the choice of ϕ and its shape parameters will be discussed in the next section.

5.2 Numerical Results

In this section, the collocation method described above is tested on several biological models. For serious implementation, the implicit ODE systems (5.3) are integrated using the subroutine `lsodi`, which is designed to solve the initial value problem for linear implicit systems of first order ODEs like (5.3), provided in the software package LSODE (Livermore Solver for Ordinary Differential Equations) [69]. LSODE uses multistep Adams/BDF methods

on nonstiff/stiff systems, where linear systems derived from implicit systems/methods are solved by direct methods (LU factor/solve), which limits its applicability to moderately large ODE systems. For very large ODE systems, software packages like LSODPK [42] or VODPK [8], both of which use iterative preconditioned Krylov methods to solve large scale linear systems, are more suitable. However, most numerical experiments presented here can also be done on a personal computer using Matlab.

5.2.1 Toggle Switch Model

The two-dimensional toggle switch model, which also has the bistable property like the Schlögl model, is a simple regulatory element in gene regulatory networks and has been used in many places ([20, 23, 40, 47, 54] to list a few) as a test problem for numerical methods. One set of relevant reaction equations can be described as

$$\left.\begin{array}{lcl} \emptyset & \xrightarrow{\alpha/(\beta+y^2)} & x \\ x & \xrightarrow{\mu_1} & \emptyset \\ \emptyset & \xrightarrow{\gamma/(\delta+x^2)} & y \\ y & \xrightarrow{\mu_2} & \emptyset \end{array}\right\}$$

with parameters $\alpha = \gamma = 10^3$, $\beta = \delta = 6 \times 10^3$, and $\mu_1 = \mu_2 = 10^{-3}$ [23].

In the collocation method, the bin size is 5×5 and $\phi(r) = e^{-(r/5)^2}$, 100 SSA runs were done starting from $x(0) = 60$, $y(0) = 10$, and $N = 530$. Figures 22 shows the evolution of the joint probability distribution in time. The bistable property is clearly evident. Without any approximation, the chemical master equation is an ODE system on the order of 10^4. However, the size of the implicit ODE system (5.3) in the collocation method is on the order of 10^2, much smaller than the original ODE system. The approximation errors for marginal probability vectors in the vector norm $\|\cdot\|_1$ are reported in Table 8(A).

Next, a stiffer system is tested using the same problem except setting $\gamma = 10^5$ and $\mu_2 = 10^{-1}$, which makes the reaction rates for reactant y about 100 times quicker than for reactant x and the SSA simulations much slower; however, the bistable phenomenon still exists. Two different bin sizes are tested here with their results shown in Figure 23, compared with the simulation result from 10^4 SSA runs. 100 SSA runs were used for the

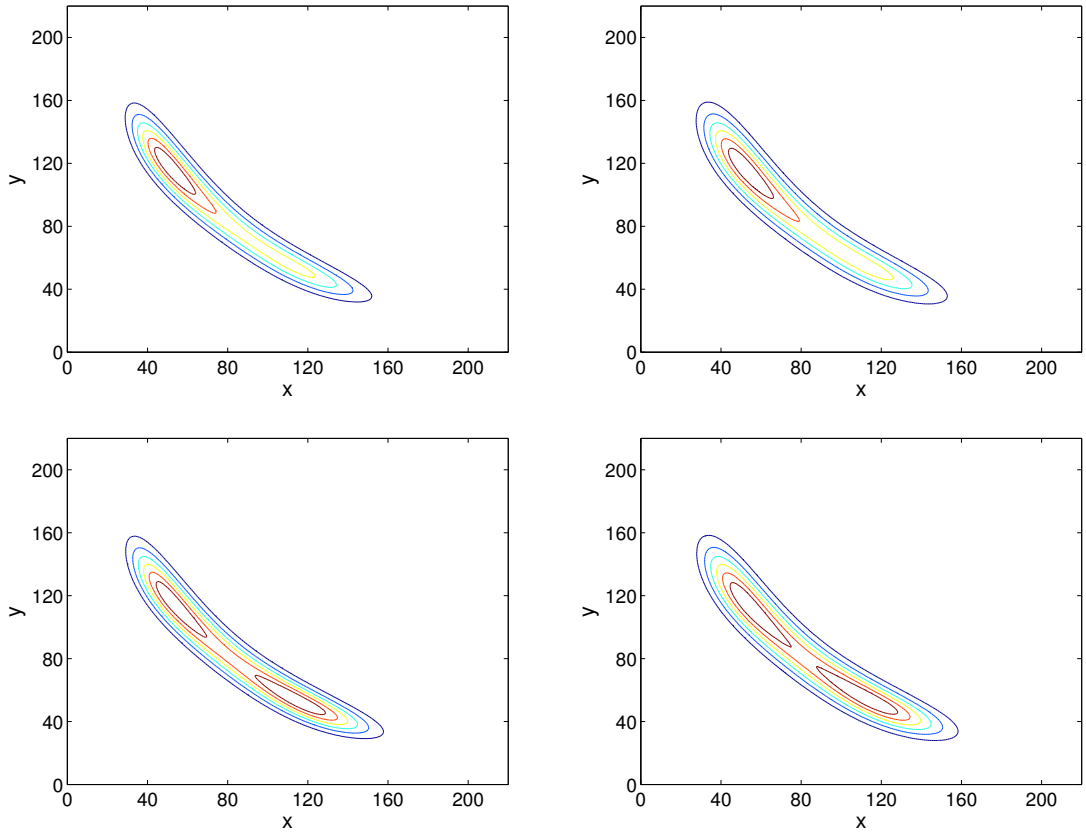


Figure 22. Contour plots of the true result (left) and the collocation result (right) at the time $t_f = 5 \times 10^4$ (top) and $t_f = 2 \times 10^5$ (bottom) respectively.

(A) Nonstiff

t_f	x	y
5^4	2.66×10^{-2}	3.36×10^{-2}
2×10^5	2.43×10^{-2}	2.43×10^{-2}

(B) Stiff

Bin sizes	x	y
5×5	0.1587	0.1453
10×5	0.07	0.0517

Table 8. Approximation errors for marginal probability vectors in the vector norm $\|\cdot\|_1$ for the toggle switch problem.

collocation method to select collocation points, resulting in $N = 533$ for the 5×5 case and $N = 284$ for the 10×5 case ($\phi(r) = e^{-(r/8.77)^2}$) and the approximation errors are reported in Table 8(B). The computation time for the collocation method is less than 30 seconds (12

seconds for the ODE solver and 15 seconds for the SSA runs), while 10^4 SSA runs took 27 minutes.

5.2.2 Cascade Model

A cascading process usually occurs when adjacent genes produce proteins that enhances the expression of succeeding genes. A simple three-species cascade process model is

$$\left.\begin{array}{llllll} \emptyset & \stackrel{\alpha}{\rightarrow} & x, & x & \stackrel{\delta}{\rightarrow} & \emptyset \\ \emptyset & \stackrel{\frac{\beta x}{\beta x+\gamma}}{\longrightarrow} & y, & y & \stackrel{\delta}{\rightarrow} & \emptyset \\ \emptyset & \stackrel{\frac{\beta y}{\beta y+\gamma}}{\longrightarrow} & z, & z & \stackrel{\delta}{\rightarrow} & \emptyset \end{array}\right\}$$

with parameters $\alpha = 0.7$, $\beta = 1.0$, $\gamma = 5.0$, and $\delta = 0.07$ [40], where species X enhances the production of species Y, and in turn species Y enhances the production of species Z. Their decay rates are assumed to be the same constant δ.

Table 9 compares the means and variances computed from the collocation method using the bin size $2 \times 2 \times 2$ and $\phi(r) = e^{-(r^2/3)}$ (100 SSA runs were conducted and $N = 388$) and 50000 SSA runs. Their computation times are nearly the same, less than 2 seconds.

Note that the starting conditions for the CME and SSA are a bit different. For the SSA, each simulation run starts from the same initial condition $(x, y, z)(0) = (4, 0, 0)$, which means that the initial probability distribution is a Kronecker delta function $p\big((4, 0, 0), 0\big) = 1$, and $p\big((x, y, z), 0\big) = 0$ if $(x, y, z) \neq (4, 0, 0)$. However, it is difficult to closely approximate such functions by Gaussian RBFs. In practice, the initial probability distribution for the collocation method is a Gaussian RBF centered at $(4, 0, 0)$.

The differences in means and variances computed from these two methods are small, especially when $t_f = 40$ where the probability distributions can be closely approximated by the Gaussian RBFs used in the collocation method. Similar observations can be made from the following examples.

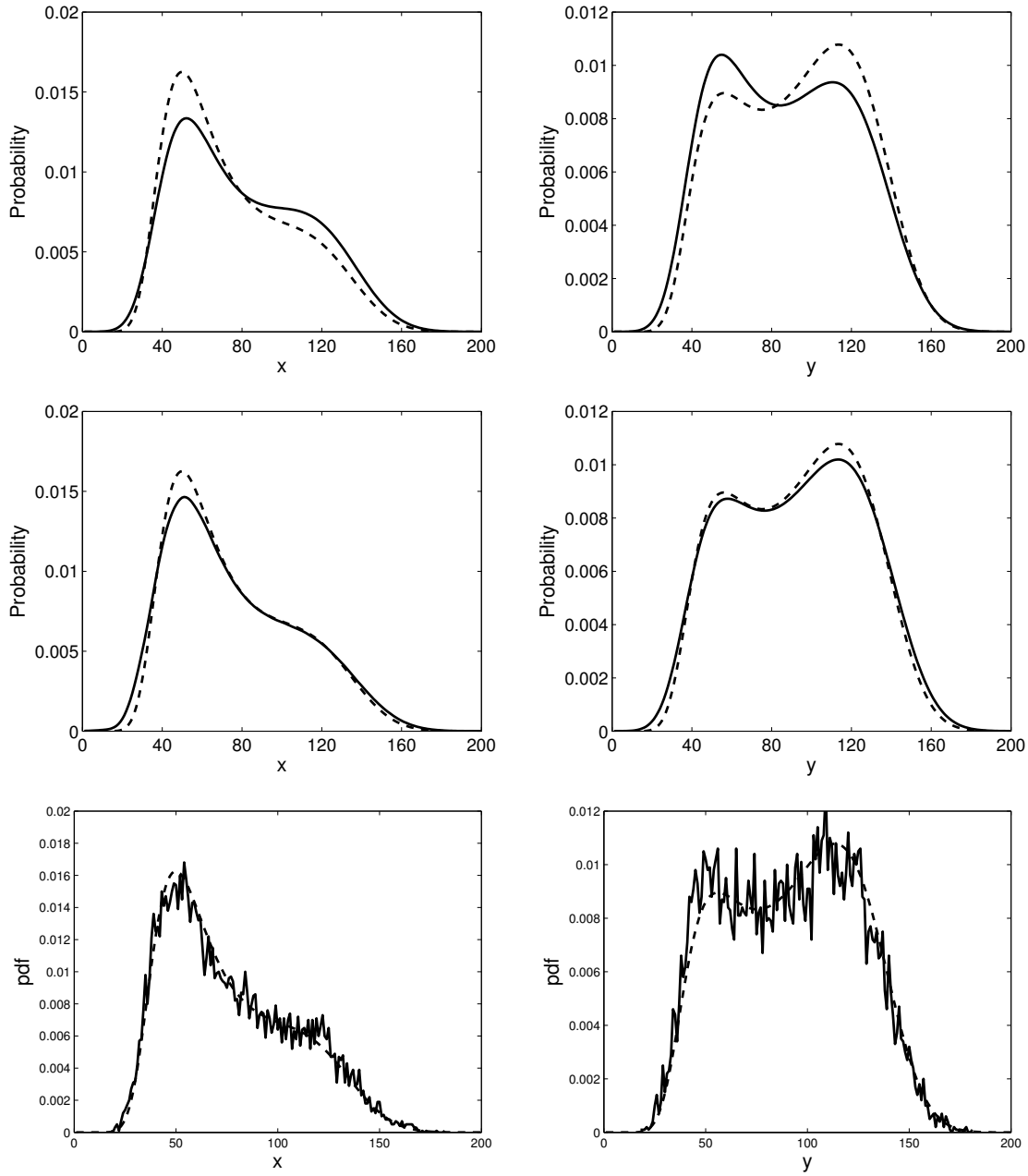


Figure 23. Marginal probability distributions for the toggle switch model of species x (left) and species y (right) at the time $t_f = 3 \times 10^4$, using the collocation method with different bin sizes 5×5 (top) and 10×5 (middle) and 10^4 SSA simulation runs (bottom), correspondingly. The dashed (solid) lines are the exact (collocation/SSA) solutions.

(A) $t_f = 10$

	CME	SSA
μ_X	7.0036	7.0540
μ_Y	3.7245	3.7563
μ_Z	1.9080	1.8975
σ_X^2	7.4570	6.1723
σ_Y^2	5.2719	4.0165
σ_Z^2	3.1785	2.4656

(B) $t_f = 40$

	CME	SSA
μ_X	9.5804	9.6560
μ_Y	8.2356	8.3063
μ_Z	7.1218	7.3506
σ_X^2	9.8361	9.8050
σ_Y^2	9.2358	9.1284
σ_Z^2	8.4976	8.3123

Table 9. Means (μ) and variances (σ^2) computed from the CME collocation method and SSA simulation runs.

5.2.3 Λ-phage

Numerical solutions to the CME for a simplified bacteriophage λ switch, which is built upon a toggle switch model between two competing proteins cI and cro, has been studied in [40] and [47]. Five different types of proteins cro, cI, N, $cIII$, cII, with their abundances denoted as x_{cro} through x_{cII}, interact with each other according to the chemical reactions

$$\left.\begin{array}{lcllll} \emptyset & \xrightarrow{\frac{\alpha_1\beta_1}{\beta_1+\gamma x_{cI}}} & x_{cro}, & x_{cro} & \xrightarrow{\delta_1} & \emptyset \\ \emptyset & \xrightarrow{\frac{(\alpha_2+kx_{cII})\beta_2}{\beta_2+\gamma x_{cro}}} & x_{cI}, & x_{cI} & \xrightarrow{\delta_2} & \emptyset \\ \emptyset & \xrightarrow{\frac{\alpha_3\beta x_{cI}}{\beta x_{cI}+\gamma}} & x_N, & x_N & \xrightarrow{\delta_3} & \emptyset \\ \emptyset & \xrightarrow{\frac{\alpha_4\beta x_N}{\beta x_N+\gamma}} & x_{cIII}, & x_{cIII} & \xrightarrow{\delta_4} & \emptyset \\ \emptyset & \xrightarrow{\frac{\alpha_5\beta x_N}{\beta x_N+\gamma}} & x_{cII}, & x_{cII} & \xrightarrow{\delta_5} & \emptyset \end{array}\right\}$$

with parameters (α_1, α_2, α_3, α_4, α_5) = (0.5, 1.0, 0.15, 0.3, 0.3), $\beta = 1.0$, (β_1, β_2) = (0.12, 0.6), $\gamma = 1.0$, (δ_1, δ_2, δ_3, δ_4, δ_5) = (0.0025, 0.0007, 0.0231, 0.01, 0.01), $k = 1$.

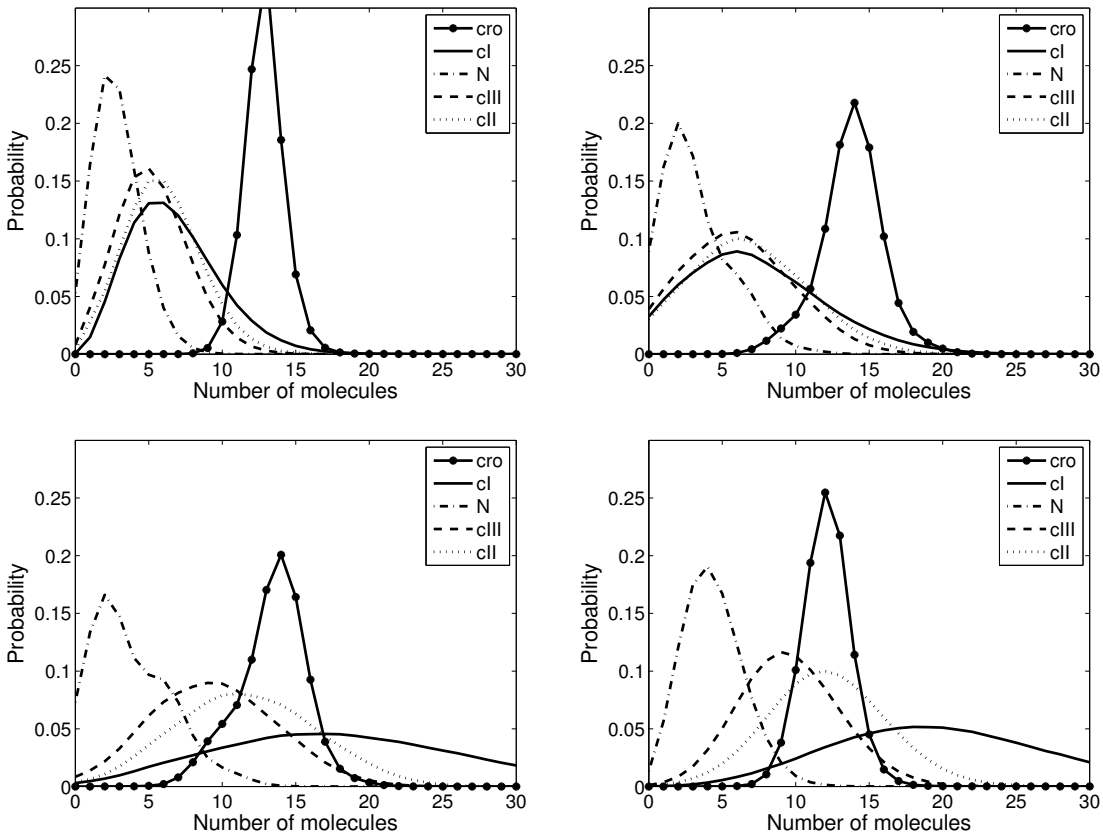


Figure 24. Marginal probability distributions for the λ phage model calculated from the SSA simulation runs (left) and the collocation method (right) at the time $t_f = 30$ (top) and $t_f = 60$ (bottom), starting from $(x_{cro}, x_{cI}, x_N, x_{cIII}, x_{cII})(0) = (14, 2, 2, 2, 2)$.

Figure 24 compares the marginal probability distributions calculated from 10^6 SSA simulation runs and the collocation method using the bin size 4^5 and $\phi(r) = e^{-(r^2/9)}$ (500 SSA runs were conducted and $N = 927$). The computation times for both methods are about the same. Note that the starting conditions are also different here, a Kronecker delta function for the SSA and a single Gaussian RBF for the CME. The numerical results from the collocation method seem reasonable. In terms of accuracy, the results are better than those in [40] and worse than those in [47] (yet, here the reduction of the problem complexity is greater than that in both references), though the initial conditions are all different.

For high dimensional problems, computing marginal probability distributions from $P(t_f) = B(t_f)\Phi$ can be very expensive. However, the basic exponential identity can help.

For instance,

$$\sum_{x_2} p(x_1, x_2, t_f) = \sum_{x_2}\sum_n b_n(t_f) \exp\left(-\left(\left(\frac{x_1 - x_1^{s_n}}{c}\right)^2 + \left(\frac{x_2 - x_2^{s_n}}{c}\right)^2\right)\right)$$
$$= \sum_n b_n(t_f) \exp\left(-\left(\frac{x_1 - x_1^{s_n}}{c}\right)^2\right) \sum_{x_2} \exp\left(-\left(\frac{x_2 - x_2^{s_n}}{c}\right)^2\right).$$

If $x_2^{s_n}$ is not on the edge of the state space and the exponential function $\exp\left(-\left(\frac{\cdot - x_2^{s_n}}{c}\right)^2\right)$ is relatively small on the edge of the state space for all n, then the summation $\sum_{x_2} \exp\left(-\left(\frac{x_2 - x_2^{s_n}}{c}\right)^2\right)$ is approximately independent of $x_2^{s_n}$, yielding the approximation

$$p(x_1, \cdot, t_f) \approx C \sum_n b_n(t_f) \exp\left(-\left(\frac{x_1 - x_1^{s_n}}{c}\right)^2\right)$$

for the marginal probability $p(x_1, \cdot, t_f)$, where C is a scaling constant. This approximation approach for computing marginal probability distributions was effective on the λ-phage model. More importantly, without approximation approaches like this, computing marginal probability distributions from partial summations of $P(t_f)$ takes much longer than computing $B(t_f)$ from the collocation method alone.

5.2.4 Goutsias Model of Regulated Transcription

Goutsias model [38], which represents a simplified component of the λ-phage virus, has been studied in [54]. The model contains ten chemical reactions and six species:

$$\left.\begin{array}{cccccc}
x_{RNA} & \xrightarrow{c_1} & x_{RNA} + x_M, & x_M & \xrightarrow{c_2} & \emptyset \\
x_{DNA.D} & \xrightarrow{c_3} & x_{RNA} + x_{DNA.D}, & x_{RNA} & \xrightarrow{c_4} & \emptyset \\
x_{DNA} + x_D & \xrightarrow{c_5} & x_{DNA.D}, & x_{DNA.D} & \xrightarrow{c_6} & x_{DNA} + x_D \\
x_{DNA.D} + x_D & \xrightarrow{c_7} & x_{DNA.2D}, & x_{DNA.2D} & \xrightarrow{c_8} & x_{DNA.D} + x_D \\
x_M + x_M & \xrightarrow{c_9} & x_D, & x_D & \xrightarrow{c_{10}} & x_M + x_M
\end{array}\right\}$$

with Avogadro's number $A = 6.0221415 \times 10^{23}$, fixed cell volume $V = 10^{-15}L$, rate constants $c_1 = 0.043$, $c_2 = 0.0007$, $c_3 = 0.0715$, $c_4 = 0.0039$, $c_5 = \frac{0.012 \times 10^9}{AV}$, $c_6 = 0.4791$, $c_7 = \frac{0.00012 \times 10^9}{AV}$, $c_8 = 0.8765 \times 10^{-11}$, $c_9 = \frac{0.05 \times 10^9}{AV}$, $c_{10} = 0.5$.

Figure 25 compares the marginal probability distributions calculated from 10^4 SSA simulation runs (0.373 seconds) and the collocation method using the bin size 1^6 and $\phi(r) = e^{-(2.0r^2)}$ (0.224 seconds), with 100 SSA runs conducted and $N = 161$. The observation is similar to that for the previous example. The proposed method uses less computational resources (memory and computation time) than that of [54], however the computational results are not quite as accurate as those shown in [54].

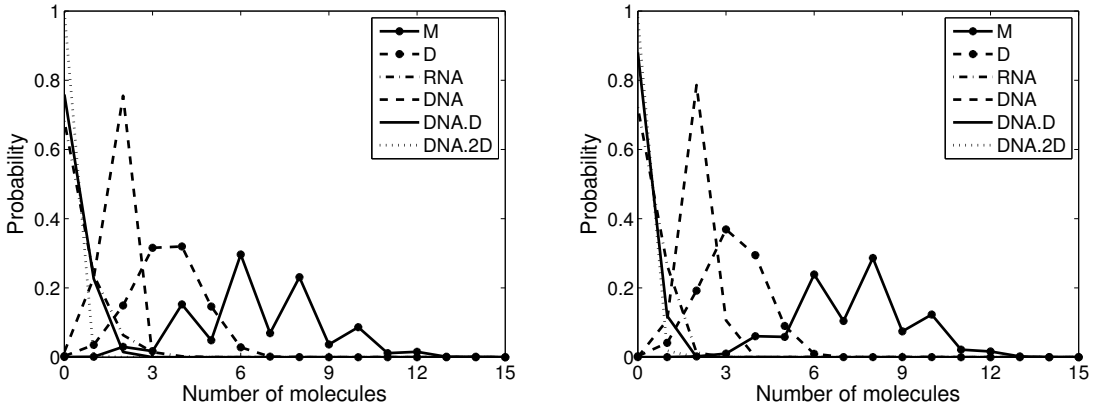


Figure 25. Marginal probability distributions for the Goutsias model calculated from the SSA simulation runs (left) and the collocation method (right) at the time $t_f = 25$, starting from $(x_M,\, x_D,\, x_{RNA},\, x_{DNA},\, x_{DNA.D},\, x_{DNA.2D})(0) = (2,\, 6,\, 0,\, 2,\, 0,\, 0)$.

5.2.5 Prototypical Cell Cycle Model

This section focuses on a prototypical cell cycle model derived from some normalized phenomenological rate equations [94], which has also been studied in [102], namely chapter 4 using a relevant method. The model can be described by the following elementary reactions without intermediates and with variable propensity rates and parameters

$$\left.\begin{array}{rcl}
\emptyset & \overset{k_1 m V_s}{\underset{k_2'}{\rightleftharpoons}} & x_{CycB} \\
x_{CycB} + x_{Cdh1} & \overset{k_2''/V_s}{\longrightarrow} & x_{Cdh1} \\
x_{CycB} + x_{Cdh1} & \overset{\frac{k_4}{J_4 V_s + x_{Cdh1}}}{\longrightarrow} & x_{CycB} + x_{Cdh1P} \\
x_{Cdc20} + x_{Cdh1P} & \overset{\frac{k_3''}{J_3 V_s + x_{Cdh1P}}}{\longrightarrow} & x_{Cdc20} + x_{Cdh1} \\
\emptyset & \overset{k_5' V_s + k_5'' V_s \frac{x_{CycB}^2}{(J_5 V_s)^2 + x_{CycB}^2}}{\underset{k_6}{\rightleftharpoons}} & x_{Cdc20}
\end{array}\right\}$$

where $k_1/c_{CycB} = 0.01\,\mathrm{min}^{-1}$, $k_2' = 0.04\,\mathrm{min}^{-1}$, $k_2'' c_{Cdh1} = 1.0\,\mathrm{min}^{-1}$, $k_3'' c_{Cdc20}/c_{Cdh1} = 1.0\,\mathrm{min}^{-1}$, $k_4 c_{CycB}/c_{Cdh1} = 3.5\,\mathrm{min}^{-1}$, $k_5'/c_{Cdc20} = 0.005\,\mathrm{min}^{-1}$, $k_5''/c_{Cdc20} = 0.2\,\mathrm{min}^{-1}$, $k_6 = 0.1\,\mathrm{min}^{-1}$, $J_3/c_{Cdh1} = 0.04$, $J_4/c_{Cdh1} = 0.04$, $J_5/c_{CycB} = 0.3$, where c_S is the characteristic concentration of the species S and $c_{CycB} = c_{Cdc20} = c_{Cdh1} = 1.0$ here. The cell mass variable m is introduced to reflect the assumption that the species $CycB$ is synthesized at a supralinear rate such that its molecular number increases with cell

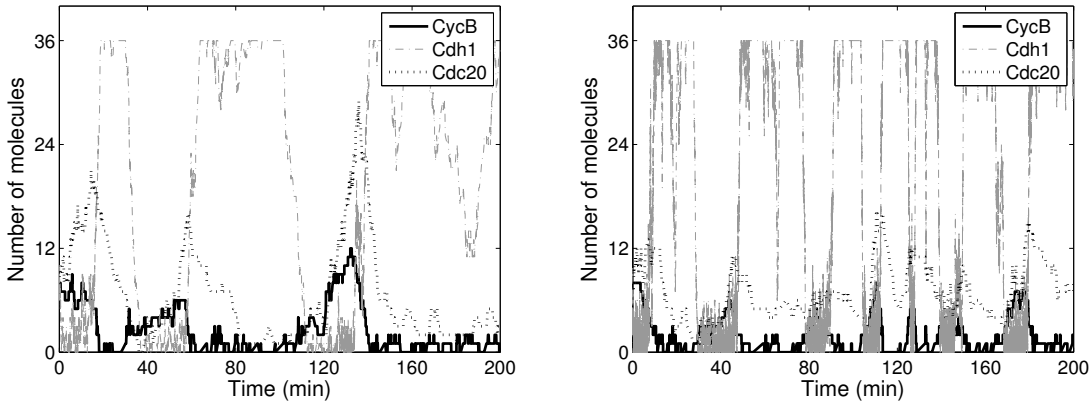


Figure 26. One SSA simulation run on time interval [0 200min] of all three species in the nonstiff toy cell cycle model (left) and the stiff one (right), starting from $(x_{CycB}, x_{Cdh1}, x_{Cdc20})(0) = (8, 8, 8)$.

mass and V_s, the nominal volume of the cell times Avogadro's number, chosen to be 18 molecules/nMolar. The species $Cdh1P$ is the phosphorylated form of $Cdh1$, therefore $x_{Cdh1} + x_{Cdh1P} = c_{Cdh1} \cdot V_s$.

Figure 27 shows comparisons between marginal probability distributions computed from the exact solution and computed from the collocation solution, using the bin size $3 \times 3 \times 3$ with $\phi(r) = e^{-(r^2/4)}$, 500 SSA runs were done and $N = 832$.

This is an example where the collocation method has trouble reproducing the probability distributions. The numerical results using the collocation method are also very sensitive to the shape parameter c in the radial function $\phi(r) = e^{-(r/c)^2}$. The shape of the joint probability mass function is not well approximated by a few translations of one single radial function $\phi(r)$.

Now, similar to the toggle switch problem, a stiffer system is tested using the same problem except setting $k_3'' c_{Cdc20}/c_{Cdh1} = 10.0\,\mathrm{min}^{-1}$, $k_4 c_{CycB}/c_{Cdh1} = 35.0\,\mathrm{min}^{-1}$, which makes the reaction rates for species C_{Cdh1} 10–100 times faster than for the other two species and the cell cycle about half (indicated in Figure 26).

Figure 28 shows comparisons between marginal probability distributions computed from the collocation solution, using different bin sizes (and different numbers N of radial functions). For all three cases, 100 SSA runs were done resulting in $N = 4776$ for the bin size $1 \times 1 \times 1$, $N = 1159$ for $2 \times 2 \times 2$, and $N = 972$ for $4 \times 1 \times 3$.

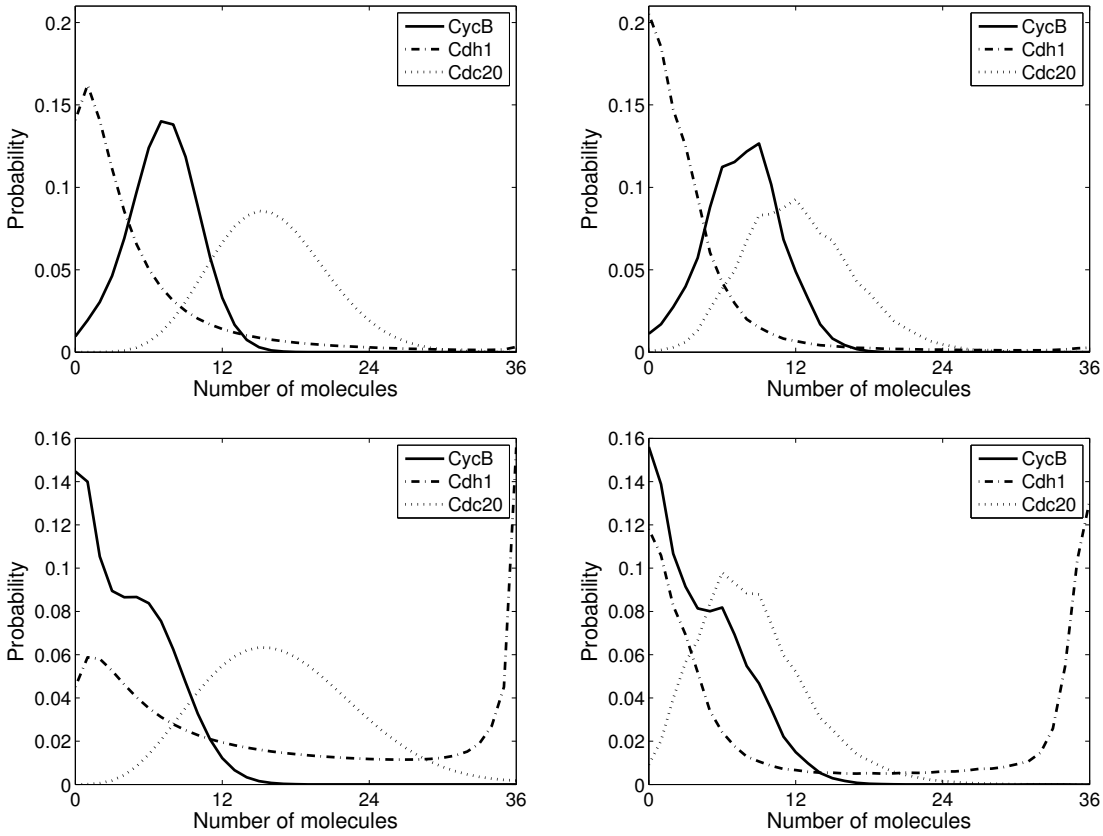


Figure 27. Marginal probability distributions for the toy cell cycle model at the time $t_f = 5\,\text{min}$ (top) and $t_f = 10\,\text{min}$ (bottom), from the exact solution (left) and the collocation solution (right).

The comparison result is similar to the result from the toggle switch model: smaller bin size (in volume) does not necessarily produce a better result. In the toggle switch case, the collocation method with bin size 10×5 produces a better approximation than with bin size 5×5, while in this case, the collocation method with bin size $4 \times 1 \times 3$ works much better than with bin size $2 \times 2 \times 2$; the latter one totally fails to track the movement of the peak of the marginal probability distribution of $Cdh1$ from low values to high values. In both cases, the collocation method solves stiff problems well if the (radial) basis functions are placed more densely (i.e., smaller edge-length of the bins) for fast reactants. The placement of the center points of the radial functions seems more important than the shape of the radial functions.

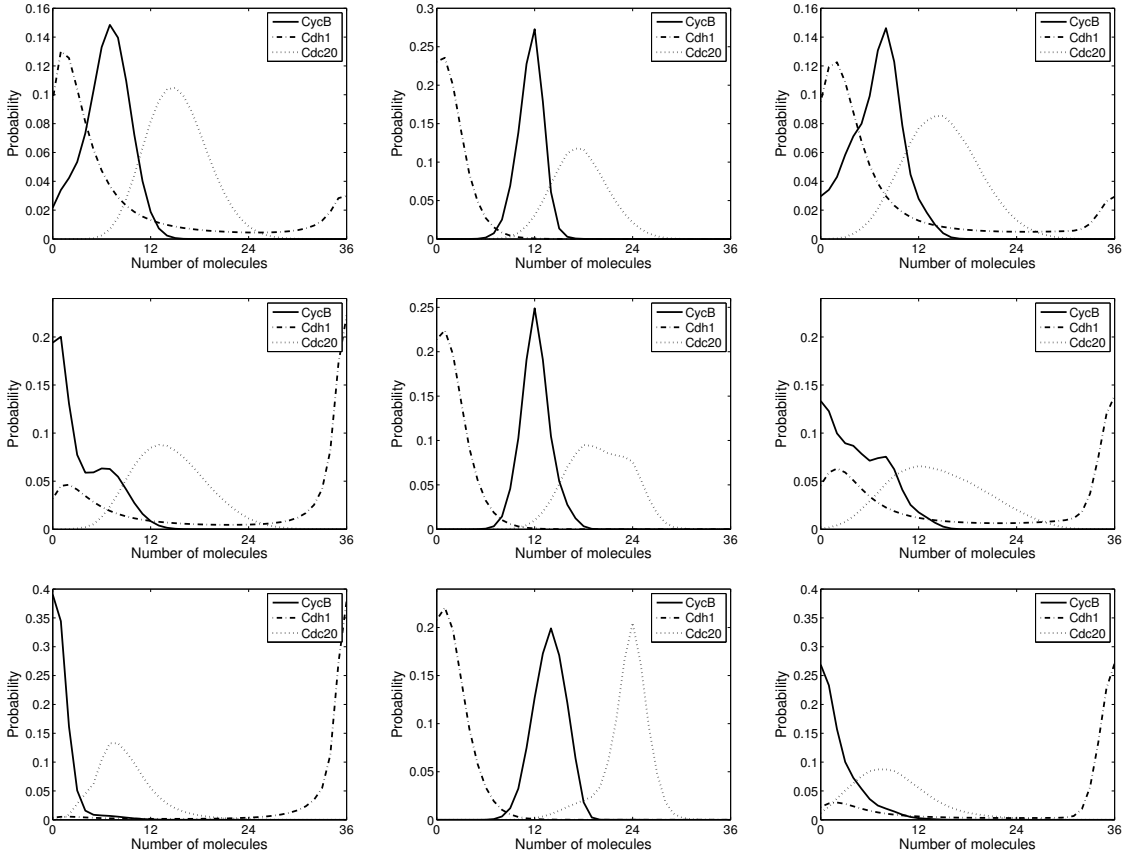


Figure 28. Marginal probability distributions for the toy cell cycle model at the time $t_f = 5\,\text{min}$ (top), $t_f = 10\,\text{min}$ (middle), and $t_f = 20\,\text{min}$ (bottom) from the collocation solution, using different bin sizes $1 \times 1 \times 1$ (left), $2 \times 2 \times 2$ (middle), and $4 \times 1 \times 3$ (right).

5.3 Numerical Analysis

The general idea of the numerical method proposed here is to try to ,approximate the solution function $p(x,t)$ to the CME as a finite sum with variables separated, like

$$p(x,t) \approx \sum_{n=1}^{N} b_n(t) F_n(x).$$

For a real variable x, common choices for the $F_n(x)$ are trigonometric functions, Legendre polynomials, Chebyshev polynomials, or radial functions. To test if the full system can be approximated closely enough by a reduced order model over the linear space spanned by $\{F_1(x),\, F_2(x),\, \ldots,\, F_N(x)\}$, the system is integrated in time using some small time step Δt and snapshots taken at times Δt, $2\Delta t$, $\ldots$, $L\Delta t$. These snapshots are stored as an $\hat{N} \times L$ matrix W, where $\hat{N}$ is the size of the full CME, the ith column of W is $\big(P(i\Delta t) - P_s\big)^T$,

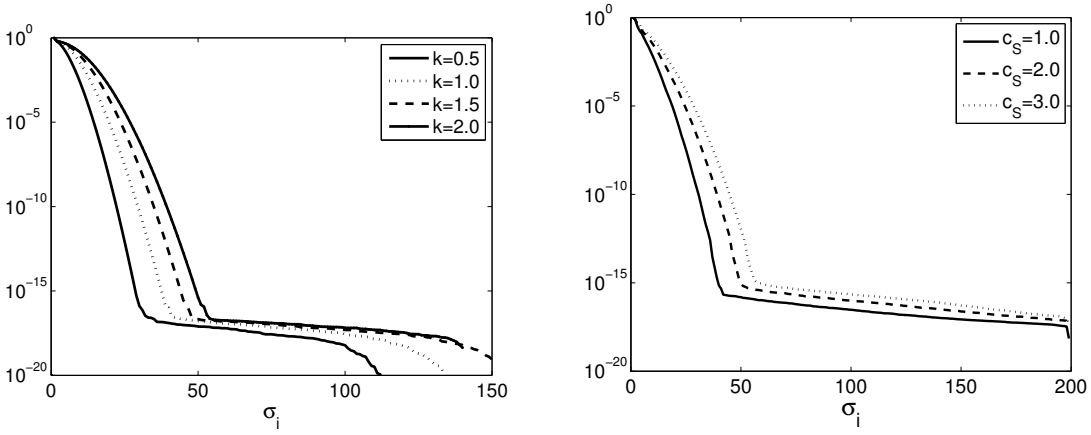


Figure 29. Scaled singular values σ_i/σ_1 of W from the birth-death model (left) and the prototypical cell cycle model (right).

the vector $P(i\Delta t)$ is the transient solution to the CME at time $i\Delta t$, and the vector P_s is the steady state solution computed from the linear system $P_s A = 0$. Typically, the size of the state space $\hat{N}$ is much larger than the number of snapshots L.

The singular value decomposition of W is $W = U\Sigma V^T$, where U is an $\hat{N} \times \hat{N}$ orthogonal matrix, V is an $L \times L$ orthogonal matrix, and Σ is an $\hat{N} \times L$ diagonal matrix with $r = \min\{\hat{N}, L\}$ nonnegative diagonal entries σ_i, arranged in decreasing order $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$, called the singular values of W.

For any $k < r$ with $\sigma_k > 0$, the matrix Σ_k constructed by replacing $\sigma_{k+1}, \sigma_{k+2}, \ldots, \sigma_r$ in Σ by zero yields an optimal rank k approximation $W_k = U\Sigma_k V^T$ to W and the approximation error $\|W_k - W\|_2 = \sigma_{k+1}$. Therefore, a good low dimensional subspace approximation to $P(t)$ exists only when a small number of singular values of W are significantly larger than the rest.

Figure 29 contains semilogarithmic plots of scaled singular values 1, σ_2/σ_1, σ_3/σ_1, ..., σ_r/σ_1 of the matrix W. The left one is for the birth-death model, where a single species X is produced at a constant rate k and diminished at a rate μ times the total number of molecules simultaneously. Starting from a Poisson distribution $p(x,0) = a_0^x e^{-a_0}/x!$, the probability distribution at time t is given by $p(x,t) = a(t)^x e^{-a(t)}/x!$, where $a(t) = a_0 e^{-\mu t} + (k/\mu)(1 - e^{-\mu t})$. As k increases, the steady state solution moves further to the right on the x-axis. This implies that more basis functions are required to obtain a close

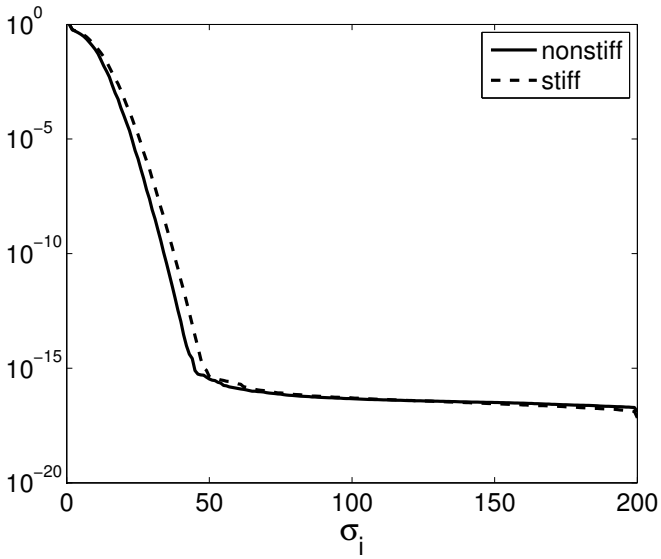


Figure 30. Scaled singular values σ_i/σ_1 of W from the toggle switch model.

approximation to $p(x,t)$, which is verified by the movement of the curves in Figure 29. This suggests that the variable separated form (5.2) might not be suitable for certain types of problems, however, other types of approximations like time-dependent wavelets might still work [46].

The next numerical experiment is on the prototypical cell cycle model. The characteristic concentration $c_{CycB} = c_{Cdc20} = c_{Cdh1} = c_S$ is increased from 1.0 to 3.0, which basically increases the problem size from $(1.0 \times V_s)^3$ to $(3.0 \times V_s)^3$ (the initial condition and the propensity functions are also changed correspondingly). Figure 29 suggests convergence of the curves as the problem size increases (the distance between the curves with $c_S = 2.0$ and $c_S = 3.0$ is a bit smaller than the distance between the curves with $c_S = 1.0$ and $c_S = 2.0$), which indicates that collocation and other basis function methods have potential for certain large scale problems.

Lastly, the toggle switch model is used to test the effect of stiffness on singular values. The result is shown in Figure 30, where the parameter setting is the same as in Section 5.2.1. To be consistent, the step size Δt for the nonstiff model is also 100 times larger than for the stiff one. Figure 30 shows little difference between the two settings, which suggests the approximation differences between stiff and nonstiff problems might only be due to the shape and location of the basis functions, not the number N of basis functions.

5.4 Conclusions

The numerical difficulty of solving the chemical master equation has been addressed by many authors, using basically grid-based approaches and Galerkin-based approaches. The new method (RBFC) proposed in this chapter can be viewed as combining of these two approaches.

From the grid-based perspective, RBFC uses a new type of radial basis function approximation scheme as the disaggregation operator. From the Galerkin-based perspective, RBFC offers more freedom in choosing the basis functions and allows the (effective numerical) support of all basis functions to be determined through SSA runs, an economical and easy way to select basis functions. On the other hand, this also makes the proposed method dependent on the SSA and the ability of Monte Carlo simulation to detect rare events (because of such dependency, there is no a priori error estimation available now). However, the use of radial basis functions makes the method more practical and flexible in higher dimensions than traditional methods that use orthogonal polynomials as basis functions. The numerical examples demonstrate the effectiveness of the new method.

The success of spectral methods depends on the choice of basis functions and collocation points. With a suitable choice guided by problem class and computational experience, one may obtain a very efficient method. For the chemical master equation, there are many ways to obtain enough information to make such a choice. For example, the steady state solution P_s to the master equation, which is relatively easy to get and only involves solving a sparse linear system $P_s A = 0$, may be used to estimate the shape parameter or to check the quality of the approximation. The moments of $p(x,t)$ in the space variables are also easy to compute and could be used for parameter determination and/or error estimation. The basis can be translates of several different functions φ_k, where each φ_k has a different shape parameter c_k, e.g., $\varphi_k(x) = \phi_k(\|x\|_2) = e^{-c_k\|x\|_2^2}$. Another avenue for improvement is to let the shape parameter c change over time, as in [22] where the orthogonal (Charlier) polynomials change shape from time to time according to the derivatives of the computed expectation values. All these and numerical experiments on higher dimensional problems (e.g., a full model of phage λ-infected *E. coli* cells [2] and a realistic budding yeast cell cycle model [64]) merit further exploration.

Chapter 6: ROBUST LINEAR SHEPARD ALGORITHM AND RIPPLE

6.1 Introduction

Interpolation problems arise in many areas where there is a need to construct a continuous surface from irregularly spaced data points. The problem is to find a surface that approximates a function defined in m-dimensional Euclidean space E^m, from a finite set of data points. Currently, there are a number of solutions to the scattered data interpolation problem. The choice of the interpolation technique depends on the distribution of points in the data set, application domain, approximating function, or the method that is prevalent in the discipline.

Shepard's interpolation method, based on a weighted average of values at the data points, usually creates good approximations. There are several variations of the original Shepard algorithm based on quadratic, cubic, linear, and trigonometric polynomials. Quadratic and cubic Shepard's method variations require more coefficients, and hence more data, than the linear Shepard method requires. Thus, in higher dimensions, it is more practical to use the linear Shepard method. Even though the performance of the linear Shepard method is comparable to other Shepard's techniques, there are situations where statistically robust least squares fits are necessary. Applications of such robust local approximations are well known in image processing. Shepard's interpolation method, together with other interpolation methods, could be used to build underlying basis functions for Galerkin-based approaches for the CME. The radial basis functions in Chapter 5 is an example of such application.

The software package SHEPPACK contains five different Fortran 95 implementations of the modified Shepard algorithm. QSHEP2D, QSHEP3D, and QSHEPMD are quadratic Shepard methods and CSHEP2D is a cubic Shepard method. LSHEP is a linear Shepard interpolation method for arbitrary dimensional data. Note that the new code LSHEP is the only one of the five that is applicable to data of dimension $m > 5$. The code LSHEP includes, as an option, a statistically robust algorithm. SHEPPACK also includes a new hybrid robust piecewise linear estimation algorithm called RIPPLE (residual initiated polynomial-time

piecewise linear estimation) intended for data from piecewise linear functions in arbitrary dimension m.

The remaining sections of this chapter are organized as follows. Section 6.2 mentions some common interpolation techniques. Section 6.3 presents Shepard's original algorithm and its modified polynomial variations. Section 6.4 presents a statistically robust linear Shepard algorithm and the motivation for a hybrid robust estimation algorithm. Section 6.5 presents the new hybrid robust algorithm RIPPLE for piecewise linear data. Section 6.6 presents performance results for all the algorithms contained in SHEPPACK.

6.2 Interpolation Techniques

Shepard [87] proposed an interpolation method that created a surface based on a weighted average of values at data points. The weight function was an inverse distance function of the data points. All methods of this type can be viewed as generalizations of Shepard's method. It was later found that this form of a weight function accorded too much influence to data points that were far away from the point of approximation. Franke and Neilson [27] developed a modification in which the weight function was designed to have local support and localized the overall approximation. This method is called the local modified Shepard method. Several variations of the modified Shepard algorithm have been developed.

Coons [15] proposed a method for describing free form curved surfaces of a very general kind. Following this work, NURBS (nonuniform rational B-splines) were proposed in [97]. They are a generalization of tensor product B-splines, and have several useful properties that have contributed to their popularity and wide commercial use. As in the case of ordinary B-splines, the sum of rational basis functions of NURBS is equal to unity. By varying the knots, NURBS can satisfy a variety of smoothness requirements. Using NURBS, it is possible to represent free form curves and surfaces as well as analytic surfaces (such as conics and quadrics). A large variety of shapes can be designed by changing the coordinates of the control points and the weights associated with them. See Piegl [66, 67] for a more detailed discussion of the properties of NURBS.

A DACE (design and analysis of computer experiments) model is an interpolating model based on Bayesian statistics. It uses the prior distribution mechanism in Bayesian statistics through which one applies past experiences, knowledge, and intuition when solving a problem. In DACE models, the unknown function is typically expressed as the sum of a known function and a Gaussian random function. The known function is usually a constant, which is estimated based on observed response values. The Gaussian random function is characterized by a covariance matrix that depends on a correlation function selected by the user. See [36] for a good overview of DACE. A more detailed discussion of the fundamental statistical and mathematical concepts can be found in [82], [50], [63], and [7].

The MARS (multivariate adaptive regression splines) technique adaptively selects a set of spline basis functions for approximating the response function through a forward/backward iterative approach. The algorithm partitions the input space into regions, each with its own regression equation. It then constructs a relation between the predictor variables and dependent variables (the spline approximation) from a set of basis functions that are entirely based on the regression data. In general, the technique is popular because it does not assume any particular type of relationship between predictor and dependent variables, and thus is widely applicable. See [28] for a complete description of MARS.

Another popular approximation method uses radial basis functions (RBFs). The significance and usefulness of approximation by RBFs follows from the theory developed in [84]. Micchelli [56] presents fundamental theory for approximation by RBFs. This type of approximation works well with scattered data because the interpolant only depends on the distances from the interpolation points. A popular choice for an interpolant function is a polyharmonic spline, which is derived from a strictly positive definite function. More details about the theory behind the choice of the interpolant function can be found in [93], who obtained results using the thin-plate spline radial basic functions together with linear polynomials.

For the special case of piecewise linear approximation in one dimension, recent work includes the algorithm L2WPMA of [18]. The code L2WPMA (least squares weighted piecewise linear approximation, Algorithm 863) calculates a piecewise monotonic approximation

to n univariate data points contaminated by random errors. The continuous piecewise linear interpolant consists of k (a positive integer provided by the user) monotonic linear splines, alternately monotonically increasing and monotonically decreasing.

An excellent source for modern approximation theory is [13], which describes many different approximation methods.

6.3 Shepard Algorithm

This section describes the original Shepard algorithm, and also presents the linear variations of the modified Shepard algorithm.

6.3.1 Original Shepard Algorithm

Local methods are attractive for very large data sets because the interpolation or approximation at any point can be achieved by considering only a local subset of the data. Many local methods can be characterized as weighted sums of local approximations $P_k(x)$, where the weights $W_k(x)$ form a partition of unity. In order for the overall method to be local, it is necessary that the weight functions have local support, that is, be nonzero over a bounded region, or at a limited number of the data points.

The original global inverse distance weighted interpolation method is due to [87]. All methods of this type may be viewed as generalizations of Shepard's method.

Let E^m denote m-dimensional Euclidean space, $x = (x_1, \ldots, x_m) \in E^m$, and for real w let $w_+ = \max\{0, w\}$. The scattered data interpolation problem can be defined as: given a set of irregularly distributed points $x^{(i)} \in E^m$, $i = 1, \ldots, n$, and scalar values f_i associated with each point satisfying $f_i = f\left(x^{(i)}\right)$ for some underlying function $f : E^m \to E$, look for an interpolating function $\tilde{f} \approx f$ such that $\tilde{f}\left(x^{(i)}\right) = f_i$. Assume that every m-simplex formed from the points $x^{(i)}$ is nondegenerate (has a nonempty interior).

Define an approximation to $f(x)$ by

$$\tilde{f}(x) = \frac{\sum_{k=1}^{n} W_k(x) f_k}{\sum_{i=1}^{n} W_i(x)},$$

where the weight functions $W_k(x)$ are defined in the original Shepard algorithm as

$$W_k(x) = \frac{1}{\left\|x - x^{(k)}\right\|_2^2}.$$

However, this form of the weight functions accords too much influence to data points that are far away from the point of approximation and may be unacceptable in some cases.

6.3.2 Modified Shepard Algorithm

Franke and Nielson [27] developed a modification that eliminates the deficiencies of the original Shepard's method. They modified the weight function $W_k(x)$ to have local support and hence to localize the overall approximation, and replaced f_k with a suitable local approximation $P_k(x)$. This method is called the *local modified Shepard method* and has the general form

$$\tilde{f}(x) = \frac{\sum_{k=1}^{n} W_k(x) P_k(x)}{\sum_{i=1}^{n} W_i(x)},$$

where $P_k(x)$ is a local approximant to the function $f(x)$ centered at $x^{(k)}$, with the property that $P_k\left(x^{(k)}\right) = f_k$. The choice for the weight functions $W_k(x)$ used in [73] was suggested by [27] and is of the form

$$W_k(x) = \left[\frac{\left(R_w^{(k)} - d_k(x)\right)_+}{R_w^{(k)} d_k(x)} \right]^2,$$

where $d_k(x) = \left\| x - x^{(k)} \right\|_2$ is the Euclidean distance between the points x and $x^{(k)}$, and the constant $R_w^{(k)} > 0$ is a radius of influence about the point $x^{(k)}$ chosen just large enough to include N_w points. The data around $x^{(k)}$ only influences $\tilde{f}(x)$ values within this radius.

There are several variations of the original Shepard algorithm based on polynomial and trigonometric functions for P_k.

The polynomial function P_k is written as a Taylor series about the point $x^{(k)}$ with constant term $f_k = P_k\left(x^{(k)}\right)$ and coefficients chosen to minimize the weighted sum of squares error

$$\sum_{\substack{i=1 \\ i \neq k}}^{n} \omega_{ik} \left[P_k\left(x^{(i)}\right) - f_i \right]^2,$$

with weights

$$\omega_{ik} = \left[\frac{\left(R_p^{(k)} - d_i\left(x^{(k)}\right)\right)_+}{R_p^{(k)} d_i\left(x^{(k)}\right)} \right]^2,$$

and $R_p^{(k)} > 0$ defining a radius about $x^{(k)}$ within which data is used for the least squares fit. R_w and R_p are taken by [27] as

$$R_w = \frac{D}{2}\sqrt{\frac{N_w}{n}}, \quad R_p = \frac{D}{2}\sqrt{\frac{N_p}{n}},$$

where $D = \max_{i,j} \left\| x^{(i)} - x^{(j)} \right\|_2$ is the maximum distance between any two data points, and N_w and N_p are arbitrary positive integers. The constant values for R_w and R_p are appropriate assuming uniform data density.

6.3.3 Linear Shepard Algorithm for Arbitrary Dimensional Data

The basis function $P_k(x)$ was the constant f_k in the original Shepard algorithm. Later variants used a quadratic polynomial, a cubic polynomial, and a cosine trigonometric polynomial [76] as basis functions. The primary disadvantage for large data sets is that a considerable amount of preprocessing is needed to determine the closest points and calculate the local approximation. The second order polynomial models have $(m+2)(m+1)/2$ coefficients for m design variables, therefore the number of data points required to estimate $P_k(x)$ is at least $(m+2)(m+1)/2$, which is prohibitive for a problem in which $m \gg 5$ and function values f_k are expensive. If $P_k(x)$ has degree d, the number of coefficients is $\binom{m+d}{d}$, requiring at least that many data points.

This consideration motivates the choice of $P_k(x)$ as linear, which would require only $(m+1)$ function values to be computed in order to construct the local least squares fit. Define $N_p = \min\{n, \lceil 3m/2 \rceil + 1\}$ as the number of points used in the local least squares approximation and

$$R^{(k)} = \min\left\{ r \mid \overline{B(x^{(k)}, r)} \text{ contains at least } N_p \text{ of the points } x^{(i)} \right\},$$

where $\overline{B(x,r)}$ is the closed ball of radius r with center x. Then, the radii R_p and R_w vary with k and are taken to be $R_p^{(k)} = 1.1R^{(k)}$, $R_w^{(k)} = \min\{D/2, R^{(k)}\}$. The choice of N_p derives from the statistics rule of thumb that $3m/2$ data points are required to reasonably estimate m parameters.

The linear Shepard method would choose $P_k(x)$ as

$$P_k(x) = f_k + \sum_{j=1}^{m} a_j^{(k)} \left(x_j - x_j^{(k)} \right).$$

Let $S = \left\{ i_1, i_2, \ldots, i_{N_p - 1} \right\}$ be the set of indices corresponding to the $N_p - 1$ points closest to $x^{(k)}$ (or those with the smaller indices in case of a tie) that determine the local least approximation $P_k(x)$. Their weights satisfying $\omega_{i_j k} > 0$ since $R_p^{(k)}$ is slightly larger than $R^{(k)}$.

Define the $(N_p - 1) \times m$ matrix A and $(N_p - 1)$-vector b by

$$A_{j\cdot} = \sqrt{\omega_{i_j k}} \left(x^{(i_j)} - x^{(k)} \right)^t,$$

$$b_j = \sqrt{\omega_{i_j k}} \left(f_{i_j} - f_k \right).$$

The coefficients $a^{(k)}$ of $P_k(x)$ are then the minimum norm solution of the linear least squares problem

$$\min_{a \in E^m} \| Aa - b \|_2,$$

found by the SVD factorization of A via the LAPACK subroutine DGELSS.

In the unlikely event that for some k the matrix A corresponding to the index set S is ill-conditioned, the computation proceeds using the minimum norm solution, which is possible since the least squares problem is solved via SVD factorization. In this event, an error flag is set indicating that at least one ill-conditioned least squares problem was encountered.

The choice of the blending influence radius $R_w^{(k)}$ is determined by the diameter D of the point set and the number of points used for a local linear least squares approximation; depending on the relative location of $x^{(k)}$ within the point set and the shape of the point set, either of the two expressions in the min may determine the minimum.

The subroutine LSHEP in SHEPPACK implements the linear Shepard algorithm as described above, for arbitrary dimension m. The function LSHEPVAL returns the approximate function value $\tilde{f}(x)$. If the weight function $W_k(x)$ is zero for all local approximations, LSHEPVAL returns the approximate function value defined in the original Shepard algorithm using only the $m + 1$ closest points to x (and again, if the smallest ball $\overline{B(x, r)}$ containing the closest $m + 1$ points also contains more than $m + 1$ points $x^{(i)}$, then only the $m + 1$ points $x^{(i)}$ with the lowest indices i are used).

6.4 Robustness in Linear Sherpard Algorithm

As described in the previous section, in the linear Shepard method, the coefficients of the linear polynomial $P_k(x)$ are the minimum norm solution to a linear least squares problem. Even though the performance of the linear Shepard method is comparable to other Shepard's techniques [29, 44], there are situations when statistically robust least squares fits are necessary. Applications of such local approximations are well known in image processing [5]. This section develops a robust linear Shepard algorithm, which is an option in the subroutine LSHEP.

6.4.1 Robust Linear Approximation Using M-estimation

By its inherent nature, least squares approximation allows "bad" data points to exert a disproportionate influence on the fit. A robust approximation is intended to be resistant to such deviant data points. Effectively, data points with larger residuals will have smaller weights and thereby less influence in the fit. M-estimation (maximum likelihood estimation) is a statistically robust technique that minimizes the sum $\sum \rho(r_i)$ of residuals r_i contributed by points being used in the fit.

In a robust linear fit using M-estimation, the coefficients of $P_k(x) = a(x - x^{(k)}) + f_k$ are chosen to minimize the weighted least squares error function

$$E(a) = \sum_{j=1}^{N_p-1} \rho\Big(P_k\big(x^{(i_j)}\big) - f_{i_j}\Big),$$

where the function ρ gives the contribution of each residual $r_j = P_k\big(x^{(i_j)}\big) - f_{i_j}$ to the total error $E(a)$. The number N_p here is the same as for the linear Shepard method. The function ρ should be C^1 and have the following properties: $\rho(r) \geq 0$, $\rho(0) = 0$, $\rho(r) = \rho(-r)$, and $\rho(r_s) \geq \rho(r_t)$ for $r_s > r_t > 0$.

Let ψ be the derivative of ρ. Differentiating the function $E(a)$ with respect to the coefficients a, and setting the partial derivatives to zero, produces a system of equations

$$\sum_{j=1}^{N_p-1} \psi\Big(a\Big(x^{(i_j)} - x^{(k)}\Big) + f_k - f_{i_j}\Big)\Big(x^{(i_j)} - x^{(k)}\Big) = 0.$$

Let the weight function be defined as $\omega(r) = \psi(r)/r$, $r \neq 0$ and $\omega(r) = 1$, $r = 0$, and let $\omega_j = \omega(r_{i_j})$. The system of equations can then be written as

$$\sum_{j=1}^{N_p-1} \omega_j \Big(a\Big(x^{(i_j)} - x^{(k)}\Big) + f_k - f_{i_j} \Big) \Big(x^{(i_j)} - x^{(k)}\Big) = 0.$$

Solving this system of equations is a weighted least squares problem, similar to the one in Section 6.3.3. The only difference is that now ω_j depends on the unknown solution a, which makes it a nonlinear system. The standard approach to solve this problem, called iteratively reweighted least squares (IRLS), is to compute and fix the weights ω_j for given a, then solve a *linear* system for new a, and iterate until convergence.

When the Huber minimax function [43] is used for ψ, the data points with larger residuals (outliers) will be "downweighted" proportional to the magnitude of the residual error. The weights will never be equal to zero, unless the residual is infinite. If the ψ function is redescending with the property that $\psi(r) = 0$ for $|r| > c$ like the bisquare minimax function, where c is a preselected cutoff value determined by the scale estimate of the residual error, then the data points with large residual errors are completely ignored. The scale of the residual error is estimated by the median absolute residual (MAR).

In practice, five iterations of IRLS with the Huber minimax function as ψ are applied first with all initial weights equal one. Five iterations of IRLS with the bisquare function are applied after that to accelerate the convergence, ideally downweighting outliers. Since IRLS with the bisquare ψ may diverge, $E(a)$, using the bisquare ρ function, for the estimate of a after the five Huber iterations is compared to $E(a)$ with the final a. If the latter is larger, then the estimate of a from the Huber iterations is used for $P_k(x)$. The influence radius $R_w^{(k)}$ is then adjusted such that the closed ball $\overline{B(x, R_w^{(k)})}$ only includes points with weights larger than some threshold, say 0.8. This adjustment of $R_w^{(k)}$ only for robust local approximations is empirically validated.

6.4.2 Motivation for a Hybrid Statistically Robust Estimation Algorithm

The breakdown bound [43, 62] for the M-estimator is $1/(p+1)$, where p is the number of parameters to be estimated. The value of the breakdown bound for the M-estimator is low. This means that a large number of data points are required to obtain robust estimates. Using the M-estimator with the linear Shepard method constructs better approximations than the standard linear Shepard method [44]. However, the primary advantage of the linear Shepard method is that, in m-dimensional space, it requires $\mathcal{O}(m)$ data points to construct the fit, which is no longer possible when M-estimation is used, since the number of points required by M-estimation is $\mathcal{O}(m^2)$. (If F is the fraction of points that are outliers, $(m+2)F \leq \frac{1}{(m+2)}(m+2) \Rightarrow (m+2)^2 F \leq m+2$, so $\mathcal{O}(m)$ outliers requires $\mathcal{O}(m^2)$ data points.) This computational complexity problem (the requirement for $\mathcal{O}(m^2)$ data points) is solved by using a different robust estimation technique, least median of squares (LMS) [79], which has a breakdown value of $1/2$, independent of the dimension m. Thus, LMS would require $\mathcal{O}(m)$ data points to construct a robust linear Shepard approximation. LMS achieves this optimal breakdown value by constructing all $\binom{n}{p+1}$ possible fits to subsets of $p+1$ points, where again p is the number of parameters being estimated. For some n and p, this is practical, but in general this factorial complexity is computationally untenable. This motivates the use of a hybrid statistically robust method, somewhere between M- and LMS-estimation, that requires fewer data points than the quadratic Shepard method and yet produces better approximations than the standard linear Shepard method.

6.5 RIPPLE

RIPPLE [45] is an acronym for *residual initiated polynomial-time piecewise linear estimation*, a new algorithm described next. As seen in the previous sections, as the space dimension m increases, the amount of data required for quadratic nodal functions $P_k(x)$ becomes computationally prohibitive. In general, the use of polynomials of degree < 2 is not sufficient to locally describe the behavior of highly nonlinear functions. However, in many engineering problems, the individual response functions are piecewise linear and thus can be approximated by using linear local approximations. An example of such a function is

the penalty function based fitness function used for the design of composite structures via genetic algorithms [29]. The core idea is that away from the breakpoints between the linear pieces, the standard linear Shepard approximation is adequate. Near a piecewise linear breakpoint, a robust linear estimator may be able to choose the "correct" linear piece, resulting in an accurate approximation, $\tilde{f}(x)$.

The standard linear Shepard algorithm uses weighted linear least squares to construct the local approximation. It uses all the data to construct the local fit. As a result, it does not produce good approximations near a piecewise linear function ridge as it "averages" all the facets near the ridge. When M-estimation is used, the linear Shepard method ideally picks the facet that is consistent with a majority of the nearby data. Even though this is usually better than the standard linear Shepard method, there are cases when the required majority of data points may not lie on the same facet as the point of approximation. This produces large errors, which can be reduced if the points to be used in the fit are chosen carefully, as LMS estimation would have done.

Let $S = \{i_1, i_2, \ldots, i_{N_p-1}\}$ be the set of indices as defined in Section 6.3.3. The most consistent points (not necessarily a majority) in the set S must be used for the local least squares approximation. Suppose that based on some criteria, the points in this set could be classified as 'inliers' and 'outliers' such that each inlier produces a lower absolute residual (vertical distance to the approximation) than all outliers. Intuition suggests that if a point is classified as an inlier, the points close to it have a higher probability of being classified as an inlier. However, if two points that have equal distance from an inlier are on two different facets, then the corresponding function values are "averaged", resulting in a large approximation error. Thus, it is not possible to pick the inliers based solely on the criterion of distance from the point of approximation x. Also, it is not possible to predict how many data points will be used for constructing the local approximation.

Suppose that the best set (set of points producing minimum sum of squared residuals) of the minimum number of points required to construct the fit is chosen. It is now possible to examine every other data point and determine whether it should be added to the best set, based on the residual that is produced. Thus, the problem is now reduced to finding the best

minimal set of points. The idea for choosing minimal sets of points required to construct the fit has been borrowed from the RANSAC (random sample consensus) algorithm [26], which chooses the minimal sets of points at random. If the data dimension is m, the number of parameters in $P_k(x)$ to be estimated is m, requiring at least $(m+1)$ data points to construct a linear approximation. The total number of data points is $N_p - 1$. The best minimal set lies among $\binom{N_p-1}{m+1}$ possible sets. However, in general this has exponential complexity and is therefore untenable.

The best minimal set of points can be approximated in polynomial time by using a special distance criterion that is described as follows. Define the $(N_p-1)\times(m+3)$ distance matrix D such that $D_{.1} = (i_1, i_2, \ldots, i_{N_p-1})^t$ and $D_{i,j+1}$ is the index of the point that has least Euclidean distance from the point whose index is $D_{i,j}$, for $i = 1$ to $N_p - 1$ and $j = 1$ to $m + 2$, subject to the constraint that every row of D must contain distinct indices. In case $x^{(i_s)}$ and $x^{(i_t)}$ are equidistant from $x^{(D_{i,j})}$, choose the one closer to $x^{(k)}$. If that choice also results in a tie, use index i_s (assuming $i_s < i_t$).

Compute the local approximation $P_k(x)$ using $x^{(k)}$ and different sets of $(m+1)$ distinct points $x^{(i)}$ that are picked as follows. For each row t, $t = 1$ to $N_p - 1$, in D, pick the point corresponding to index $D_{t,1}$. The remaining m points can be picked in $\binom{m+2}{m}$ ways from row t. Thus, the total number of local approximations $P_k(x)$ computed using $x^{(k)}$ and a set of $(m+1)$ other points will be $(N_p-1)\binom{m+2}{m}$. For each approximation, compute the absolute residuals (vertical distances) for the $(m+1)$ points used to compute the least squares fit. Record which set of $(m+1)$ points produces the minimum least squares error. Let $R = \{i_1, i_2, \ldots, i_{m+1}\}$ correspond to the indices of these $m+1$ points. In case of a tie for this best set, choose the set containing the points closer to $x^{(k)}$ (precisely, sort the distances $\|x^{(i_j)} - x^{(k)}\|$ into an $(m+1)$ vector in increasing order, then compare the two vectors lexicographically; if these distances are equal, sort and compare lexicographically the sets of indices).

Once the best set of minimal points is obtained, compute the linear interpolant at $x^{(k)}$. Compute the average residual produced by the points $x^{(i)}$, $i \in R$, and record the coefficients of the linear interpolant. Use the method of robust M-estimation to determine the set $T \supset R$

of points for computing the final local approximation. For solving the nonlinear system using the iteratively reweighted least squares method, use the coefficients and MAR of the linear fit obtained from set R as the initial linear least squares approximation and scale estimate, respectively. Compute the final local approximation $P_k(x) = f_k + \sum_{j=1}^{m} a_j^{(k)} \left(x_j - x_j^{(k)} \right)$ using points $x^{(k)}$ and $x^{(i)}$, $i \in T$.

The algorithm RIPPLE described above is implemented in the subroutine RIPPLE in SHEPPACK, for arbitrary dimension m, and RIPPLE uses the same function LSHEPVAL as LSHEP to return the approximate function value $\tilde{f}(x)$. RIPPLE is specifically intended for sparse scattered data from approximately piecewise linear continuous functions. In other contexts RIPPLE may perform erratically compared to, say, LSHEP.

6.6 Performance

Five continuous, piecewise smooth test functions $f_i : [0,1]^m \to E$ ($i = 1, \ldots, 5$), similar to functions occurring in many applications [5, 29], are used to test the performance of the algorithms in LSHEP. The test functions f_1, f_2, f_3, f_4, and f_5 for $m = 2$ are shown in Figures 31 (A)–(E), respectively.

The function f_1 in m dimensions is

$$f_1(x) = \begin{cases} \frac{2}{m} \sum_{i=1}^{m} x_i, & \sum_{i=1}^{m} x_i \le \frac{m}{2}, \\ -\frac{2}{m} \sum_{i=1}^{m} x_i + 2, & \sum_{i=1}^{m} x_i > \frac{m}{2}, \end{cases}$$

which has the maximum $f_1(x^*) = 1$ at $\sum_{i=1}^{m} x_i^* = \frac{m}{2}$.

The function f_2 in m dimensions is

$$f_2(x) = 1 - \frac{2}{m} \sum_{i=1}^{m} |x_i - 0.5|,$$

which has the maximum $f_2(x^*) = 1$ at $x_i^* = 0.5$, for $i = 1, \ldots, m$.

The function f_3 in m dimensions is

$$f_3(x) = 1 - 2 \max_{1 \le i \le m} (|x_i - 0.5|),$$

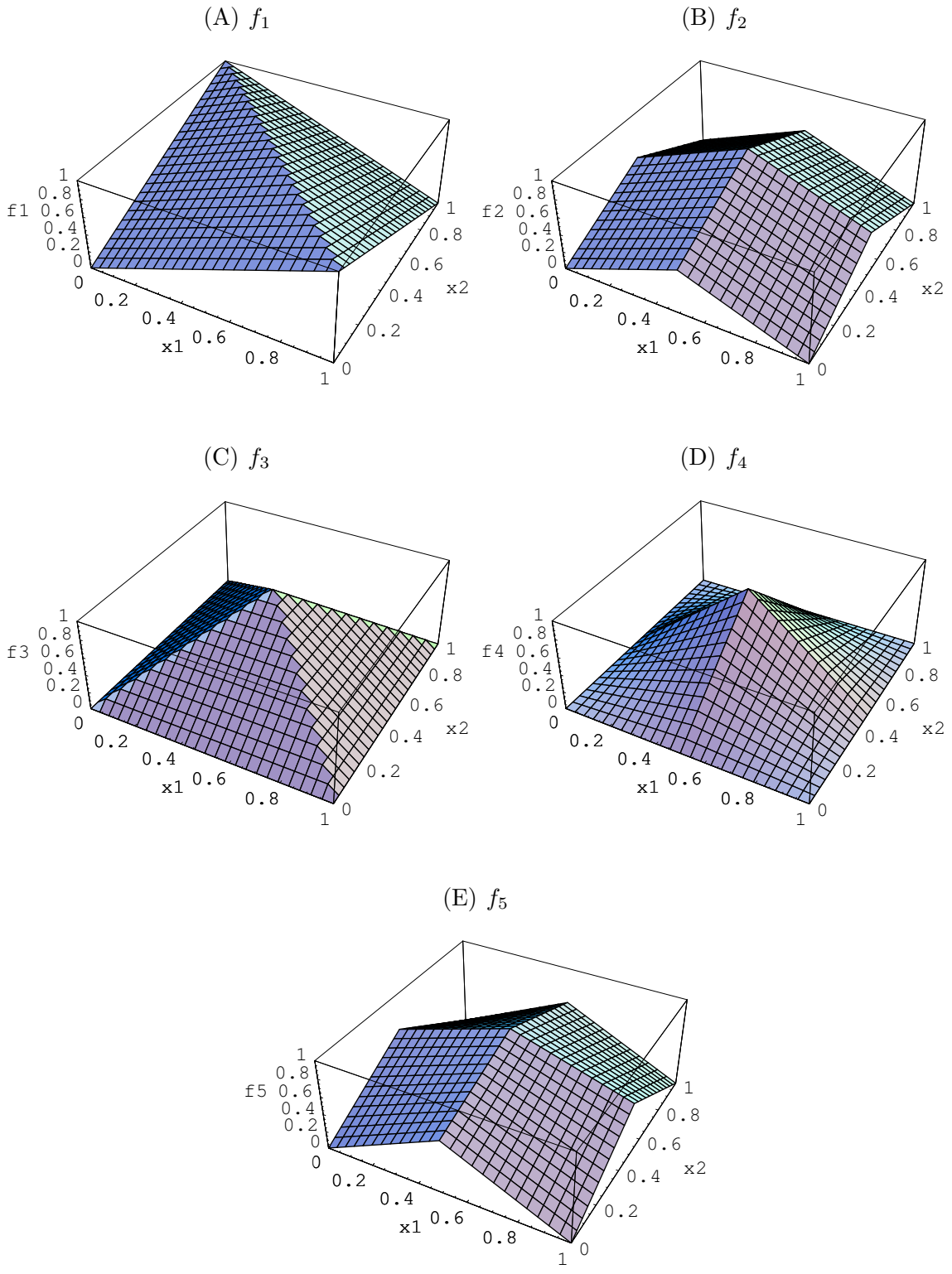


Figure 31. The test functions f_1, f_2, f_3, f_4, and f_5 for $m = 2$.

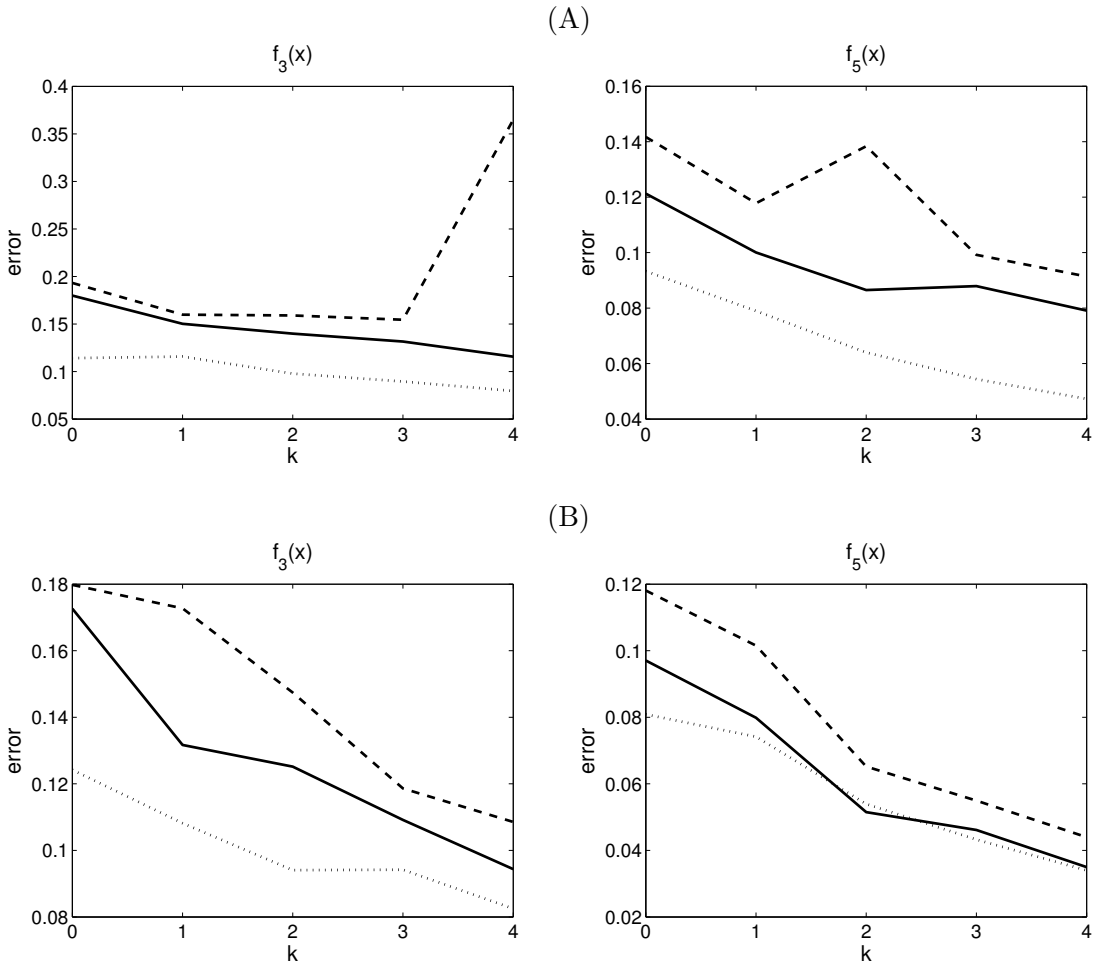


Figure 32. $f_i(x)$, $i = 3$ and 5, $m = 5$. RMS error plots for LSHEP (solid), robust LSHEP (dashed), and RIPPLE (dotted) versus $n_0 = 100 \cdot 2^k$, $n_1 = 8$. (A) Noisy data. (B) Noisy data with outliers.

which has the maximum $f_3(x^*) = 1$ at $x_i^* = 0.5$, for $i = 1, \ldots, m$.

The function f_4 in m dimensions is

$$f_4(x) = \prod_{i=1}^{m} g_i(x),$$

where

$$g_i(x) = \begin{cases} 2x_i, & x_i \le 1/2, \\ 2(1 - x_i), & \text{otherwise}, \end{cases}$$

which has the maximum $f_4(x^*) = 1$ at $x_i^* = 0.5$, for $i = 1, \ldots, m$.

The function f_5 in m dimensions is

$$1 - \frac{1}{0.5m + 0.5^m} \left(\sum_{i=1}^{m} |x_i - 0.5| + \prod_{i=1}^{m} |x_i - 0.5| \right),$$

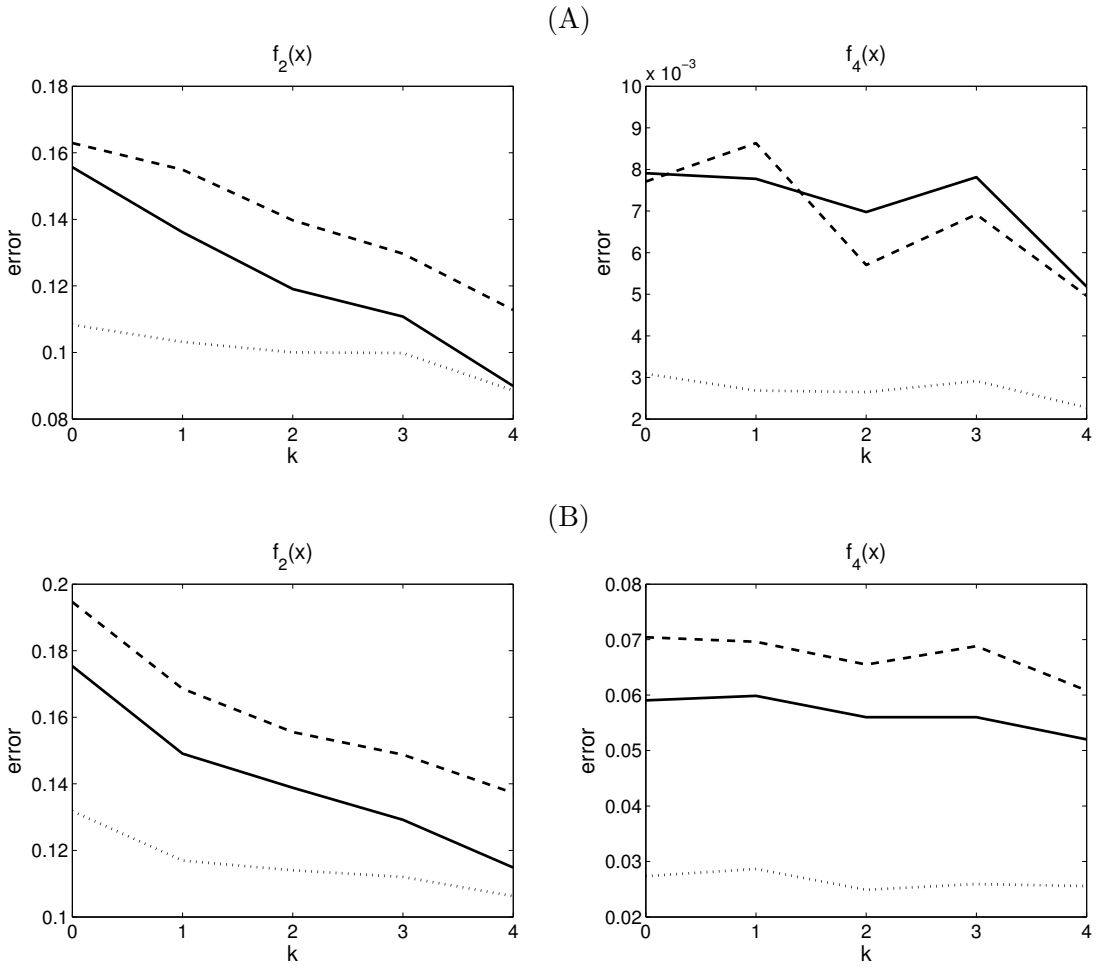


Figure 33. $f_i(x)$, $i = 2$ and 4, $m = 10$. RMS error plots for LSHEP (solid), robust LSHEP (dashed), and RIPPLE (dotted) versus $n_0 = 100 \cdot 2^k$, $n_1 = 4$. (A) Noisy data. (B) Noisy data with outliers.

which has the maximum $f_5(x^*) = 1$ at $x_i^* = 0.5$, for $i = 1, \ldots, m$.

Each test function was sampled at n_0 random (uniformly distributed) scattered points of $Q = [0,1]^m$ to generate the data set $\{x^{(i)}\}_{i=1}^{n_0}$ used to construct the Shepard interpolants. The approximation error has been characterized using three error metrics. These are the maximum absolute error $e_{\max}$, the mean absolute error $\bar{e}$, and the root mean squared error e_r. The absolute approximation error is defined as

$$e_i = \left|\tilde{f}\left(z^{(i)}\right) - f\left(z^{(i)}\right)\right|,$$

where $\tilde{f}(x)$ is the interpolant function, $f(x)$ is the test function, and $z^{(i)}$ are the points of a uniform $n_1 \times \cdots \times n_1$ grid $G \subset [0.1, 0.9]^m$. The total number of grid points is $n = n_1^m$.

Using this notation, the maximum absolute error is

$$e_{\max} = \max_{1 \le i \le n} e_i,$$

the mean absolute error is

$$\bar{e} = \frac{1}{n} \sum_{i=1}^{n} e_i,$$

and the root mean squared error is defined as

$$e_r = \sqrt{\sum_{i=1}^{n} e_i^2 \Big/ n}.$$

Figure 32 shows the interpolation errors for two test problems f_3, f_5 with $m = 5$ (five-dimensional) and the results for the three other test problems are similar. The results are only shown under the root mean square error measure, since the maximum absolute error $e_{\max}$ offers little information as $e_{\max}$ is usually determined by points $z^{(i)}$ (far from all the interpolation points $x^{(i)}$) where all local linear approximations have little or no effect, and the behavior of the mean absolute error $\bar{e}$ is very similar to e_r.

The first test is conducted with $N(0, 10^{-6})$ normally distributed noise (zero mean, standard deviation 0.001) added, with results shown in Figure 32(A). Next, to test how well outliers in the data are dealt with, contaminated function values

$$f_i(x) + 0.001\,\mu + 0.1\,\chi_{[0,0.2]}(\nu),$$

are used, where μ is a $N(0, 1)$ normally distributed random variable, ν is a $U[0, 1]$ uniformly distributed random variable, and $\chi_{[0,0.2]}$ is the characteristic function of the interval $[0, 0.2]$. The effect is to add small noise everywhere and large noise producing an outlier with probability 0.2. Note that since the Shepard approximation $\tilde{f}(x)$ interpolates all the data points $\left(x^{(i)}, f_i\right)$, if x is close to an outlier $x^{(i)}$ then $\tilde{f}(x)$ will have a large error. The goal is to have $\tilde{f}(x)$ for x away from outliers to be controlled by inliers, and thus be accurate. Although SHEPPACK does not support this, one could consider modifying the outlier data values to be consistent with the inlier data, and use a Shepard approximation based on these modified data values. This strategy, for piecewise linear functions, is called the "slope facet

model" in image processing, and is a special case of more sophisticated "variable order" facet models [55]. Figure 32(B) shows the results of the outlier test. Figure 33 is analogous to Figure 32 for two other test problems f_2 and f_4 with $m = 10$ (ten-dimensional).

In most situations, RIPPLE has less RMS error than LSHEP (the codes using statistically nonrobust least squares estimation) and the robust linear estimation option in LSHEP. However, the robust linear estimation option in LSHEP does not always improve and sometimes degrades the approximation result without the option. The reason for this is illustrated in Figure 34, which shows two simple one-dimensional examples. Plots on the left side show a case where LSHEP with the robust option could improve the approximation result, however, plots on the right show a case where the robust option makes the approximation worse.

The differences between all these are explained by the different ways weights are determined when constructing the local linear approximation P_k about each sample point $x^{(k)}$. In LSHEP without the robust option, weights are determined totally by distances between other sample points and $x^{(k)}$. However, in LSHEP with the robust option, weights are determined mostly by function values, i.e., points with function values closer to f_k tend to get larger weights. In RIPPLE large weights are credited to points that form the best local fit around $x^{(k)}$ (best in the sense of producing minimum least squares error). The numerical results show the RIPPLE strategy is the best when approximating piecewise functions in high dimensions, while the LSHEP robust one is not so good and the LSHEP nonrobust one is somewhere in between. However, RIPPLE is considerably more expensive in high dimensions than the robust LSHEP option, which performs well in the presence of a few outliers compared to LSHEP (cf. the left side of Figure 34). Thus the robust M-estimation option in LSHEP, while not generally the best choice, is still worth keeping as an option.

Figure 34 also shows an inherent problem of linear Shepard algorithms, especially for RIPPLE. In the two plots for RIPPLE in Figure 34, the local linear approximation functions are very close to the test function f_1, however, the approximation error is still quite significant around the vertex/ridge of the piecewise linear function. This happens whenever the radius

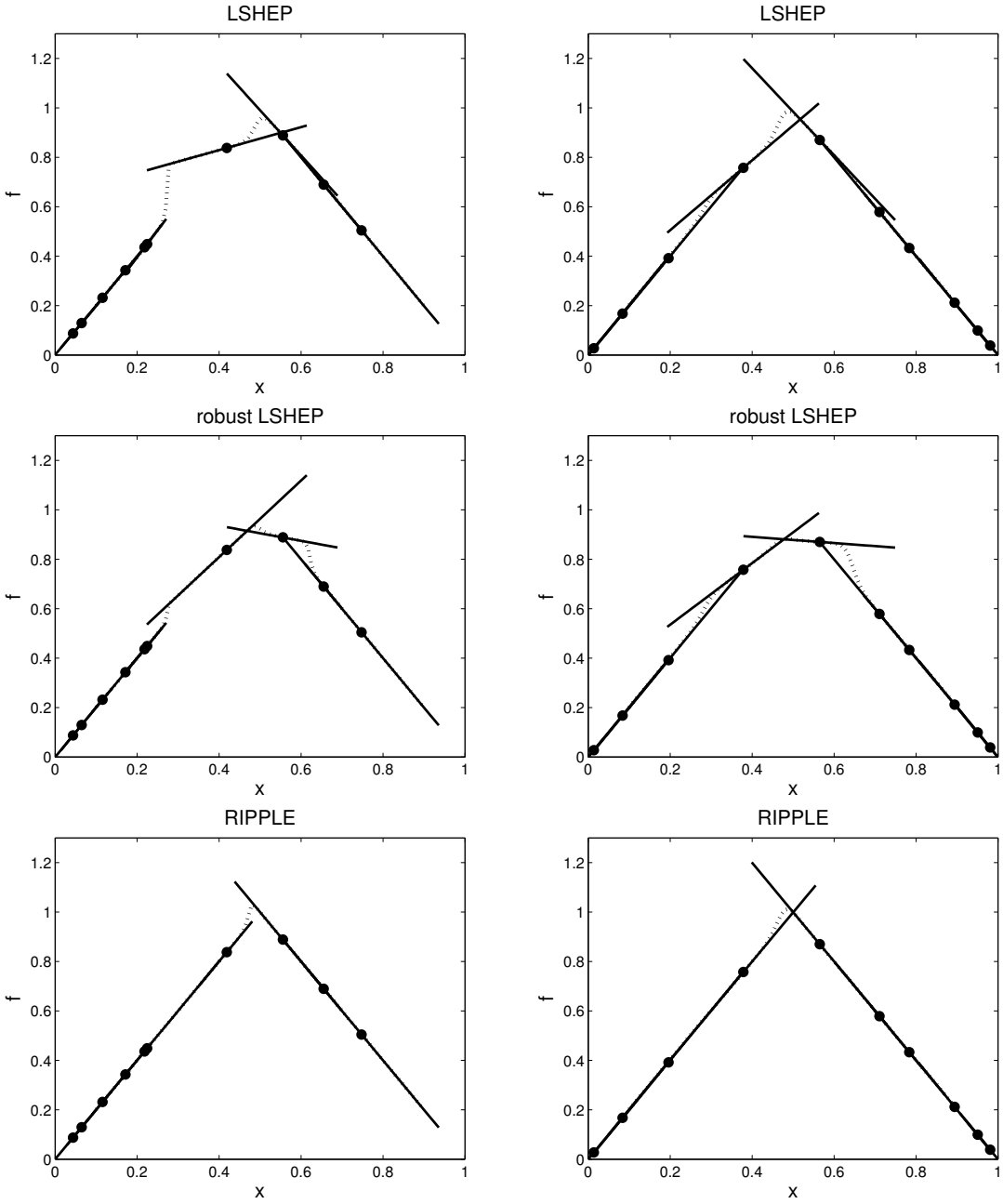


Figure 34. Local linear approximation functions and Shepard approximation functions for $f_1(x)$ using LSHEP without the robust option, with the robust option, and RIPPLE, from two different sets of sample points. The dotted nodes are the data set $\{(x^{(i)}, f_1(x^{(i)})+0.001\mu)\}_{i=1}^{n_0}$ used to build local approximations and the radius of each local approximation (solid lines) is $R_w^{(i)}$. The dotted lines are the interpolant function $\tilde{f}(x)$.

of influence of a sample point on one facet of the piecewise (linear) function extends over other facets of the function. In this case, even if the local approximation is correct, applying

the local approximation within its radius of influence still results in a large approximation error. In the new code LSHEPVAL, a warning flag is turned on if the angle between the tangent plane of $\tilde{f}(x)$ at point $z^{(i)}$ and that of some local approximation $P_k(x)$ with nonzero weight $W_k(z^{(i)})$ is over a threshold of 30°, indicating potential proximity to a ridge/vertex of a piecewise (linear) function where the approximation error could be significant.

Chapter 7: CONCLUSIONS and FUTURE WORK

This thesis concerns numerical methods for the chemical master equation, especially when applied to systems biology. The first part of the work used existing numerical algorithms on the CME, namely the differential form of the Chapman-Kolmogorov equation or the Kolmogorov forward equation for large, continuous-time, discrete-state Markov chains. Three types of methods were tested: the uniformization/randomization method and its variants, traditional single step and multistep numerical ODE solvers, and Krylov space methods. To test the scalability of these methods, large scale experiments were conducted on parallel machines.

To further reduce the computational effort to solve the CME, different approximation techniques for the CME were considered. This work introduced two such algorithms, the grid-based adaptive aggregation method and the Galerkin-based radial basis function collocation method. Numerical results for these two approaches are promising. The computational cost is significantly reduced, while the approximation errors are reasonable.

The third contribution of this thesis is the development and numerical analysis of the linear Shepard algorithm with an optional statistically robust M-estimation feature and a new hybrid robust piecewise linear estimation algorithm, RIPPLE, that interpolate arbitrary dimensional data. Interpolation techniques like the Shepard algorithm could be used as surrogate models to approximate solutions to the CME as did the radial basis functions.

Future work includes combining different surrogate models with existing numerical techniques to solve CMEs with large numbers of different species. Hybrid methods that use multilevel approximate kinetics like the Langevin and reaction rate equation with the mesoscopic chemical master equation also hold great promise. Finally, it is absolutely necessary to explore the implementation of all these ideas on parallel machines, since the ultimate goal is to understand realistic biological models containing hundreds of different reaction channels and chemical species that could only be manipulated on supercomputers.

BIBLIOGRAPHY

[1] Angus, J. E., *Some bounds on the error in approximating transition probabilities in continuous-time Markov processes*, SIAM Rev., 34 (1992) 110–113.

[2] Arkin, A., Ross, J., and McAdams, H.H., *Stochastic Kinetic Analysis of Developmental Pathway Bifurcation in Phage λ-Infected Escherichia coli Cells*, Genetics, 149 (1998) 1633–1648.

[3] Berman, A., Plemmons, R. J., *Nonnegative Matrices in the Mathematical Sciences*, SIAM, Philadelphia, PA, 1994.

[4] Berry, M. W., and Minser, K. S., *Algorithm 798: High-dimensional interpolation using the modified Shepard method*, ACM Transactions on Mathematical Software 25, 3 (1999) 353–366.

[5] Besl, P. J., Birch, J. B., and Watson, L. T., *Robust window operators*, Machine Vision and Applications, 2 (1989) 179–191.

[6] Bobbio, A., Trivedi, K. S., *An aggregation technique for the transient analysis of stiff Markov chains*, IEEE Trans. Comput., C-35 (1986) 803–814.

[7] Booker, A. J., Conn, A. R., Dennis, J. E., Frank, P. D., Trosset, M., and Torczon, V., *Global modeling for optimization: Boeing/IBM/Rice Collaborative Project*, Text provided by Paul D. Frank of Boeing, 1995.

[8] Bryne, G.D., *Pragmatic experiments with Krylov methods in the stiff ODE setting*, in Computational Ordinary Differential Equations, J.R. Cash and I. Gladwell (eds.), Oxford University Press, Oxford, UK, 1992, 323-356.

[9] Burrage, K., Hegland, M., Macnamara, S. and Sidje, R., *A Krylov-based finite state projection algorithm for solving the chemical master equation arising in the discrete modeling of biological systems*, in Proc. of The A.A.Markov 150th Anniversary meeting, (2006) 21–37.

[10] Cao, Y., Gillespie, D. T., and Petzold, L. R., *The slow-scale stochastic simulation algorithm*, J. Chem. Phys., 122 (2005) 014116.

[11] Cao, Y., Gillespie, D. T., and Petzold, L. R., *Multiscale stochastic simulation algorithm with stochastic partial equilibrium assumption for chemically reacting systems*, J. Comput. Phys., 206 (2005) 395–411.

[12] Carmo, R., de Souza e Silva, E., and Marie, R., *Efficient solutions for an approximation technique for the transient analysis of Markovian models*, Technical Report, IRISA, no. 3055, Campus universitaire de Beaulieu.

[13] Cheney, E. W., Light, W. A., *A Course in Approximation Theory*, Brooks/Cole, Pacific Grove, 2000.

[14] Ciliberto A., Capuani, F., and Tyson J.J., *Modeling Networks of Coupled Enzymatic Reactions Using the Total Quasi-Steady State Approximation*, PLoS Comput. Biol., 3 (2007) e45.

[15] Coons, S., *Surfaces for computer-aided design of space forms*, Technical Report TR-41, Massachusetts Institute of Technology, Cambridge, MA, USA, 1967.

[16] Davis, T. A., *UMFPACK version 5.0 user guide*, Technical Report, TR-04-003, Dept. of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, 2004.

[17] Davis, T. A., *Direct Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, 2006.

[18] Demetriou, I. C., *Algorithm 863: L2WPMA, a Fortran 77 package for weighted least-squares piecewise monotonic data approximation*, ACM Transactions on Mathematical Software, 33 (2007) 1–16.

[19] Deuflhard, P. and Wulkow, M., *Computational treatment of polyreaction kinetics by orthogonal polynomials of a discrete variable*, Impact Comput. Sci. Eng., 1 (1989) 269-301.

[20] Deuflhard, Huisinga, W., Jahnke, T., and Wulkow M., *Adaptive discrete Galerkin methods applied to the chemical master equation*, Technical Report, ZIB-Report 07-04, Konrad-Zuse Zentrum fuer Informationstechnik, Berlin, 2007.

[21] Engblom, S., *A discrete spectral method for the chemical master equation*, Technical Report 2006-036, Dept. of Information Technology, Uppsala University.

[22] Engblom, S., *Galerkin spectral method applied to the chemical master equation*, Commun. Comput. Phys., 5 (2008) 871–896.
[23] Engblom, S., *Spectral approximation of solutions to the chemical master equation*, Journal of Computational and Applied Mathematics, 229 (2009) 208–221.
[24] Elf, J., Lötstedt, P., and Sjöberg, P., *Problems of high dimension in molecular biology*, in Proc. 19th GAMM-Seminar, Leipzig, Germany, 2003.
[25] Ferm, L. and Lötstedt, P., *Adaptive solution of the master equation in low dimensions*, Applied Numerical Mathematics, 59 (2009) 187–204.
[26] Fischler, M. A. and Bolles, R. C., *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM, 24 (1981) 381–395.
[27] Franke, R., and Neilson, G., *Smooth interpolation of large sets of scattered data*, International Journal of Numerical Methods in Engineering, 15 (1980) 1691–1704.
[28] Friedman, J. H., *Multivariate adaptive regression splines*, Annals of Statistics, 19 (1991) 1–141.
[29] Gantovnik, V., Gürdal, Z., and Watson, L. T., *Linear Shepard interpolation for high dimensional piecewise smooth functions*, in Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY, 2004, CD-ROM, 15 pages.
[30] Gardner, T. S., Cantor, C. R., and Collins, J. J., *Construction of a genetic toggle switch in Escherichia coli*, Nature, 403 (2000) 339–342.
[31] Gibson, M. A., Bruck, J., *Efficient exact stochastic simulation of chemical systems with many species and many channels*, J. Phys. Chem., 104 (2000) 1876–1889.
[32] Gillespie, D. T., *Exact stochastic simulation of coupled chemical reactions*, J. Phys. Chem., 81 (1977) 2340–2360.
[33] Gillespie, D. T., *A rigorous derivation of the chemical master equation*, Physica A, 188 (1992) 404–425.
[34] Gillespie, D. T., *Markov Processes: an Introduction for Physical Scientists*, Academic Press, Boston, 1992.
[35] Gillespie, D. T., *Approximate accelerated stochastic simulation of chemically reacting systems*, J. Chem. Phys., 115 (2001) 1716–1733.
[36] Giunta, A. A., *Aircraft multidisciplinary design optimization using design of experiments theory and response surface modeling methods*, Ph.D. thesis, Department of Aerospace and Ocean Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1997.
[37] Goldbeter, A., Koshland, D. E., *An amplified sensitivity arising from covalent modification in biological systems*, Proc. Natl. Acad. Sci., 78 (1981) 433–457.
[38] Goutsias, J., *Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems*, J. Chem. Phys., 122 (2005) 184102.
[39] Grassmann, W. K., *Computational Probability, 1st edition*, Springer, 1999.
[40] Hegland, M., Burden, C., Santoso, L., MacNamara, S., and Booth H., *A solver for the stochastic master equation applied to gene regulatory networks*, J. Comput. Appl. Math., 205 (2005) 708–724.
[41] Hegland, M., Hellander, A., and Lötstedt, P., *Sparse grids and hybrid methods for the chemical master equation*, BIT Numerical Mathematics, 48 (2008) 265–283.
[42] Hindmarsh, A.C., *ODEPACK, a Systematized Collection of ODE Solvers*, in IMACS Transactions on Scientific Computation, R. S. Stepleman (eds.), North-Holland, Amsterdam, 1983, 55–64.
[43] Huber, P., *Robust Statistics*, John Wiley and Sons, New York, 1981.
[44] Iyer, M. A. and Watson, L. T., *An interpolation method for high dimensional scattered data*, in Proceedings of Spring Simulation Multiconference, Business and Industry Symposium, J. A. Hamilton, Jr., R. MacDonald, and M. J. Chinni (eds.), Society for Modeling and Simulation International, San Diego, CA, 2006, 217–222..

[45] Iyer, M. A. and Watson, L. T., *RIPPLE: Residual Initiated Polynomial-time Piecewise Linear Estimation*, in Proceedings IEEE Southeastcon 2007, Richmond, VA, 444–449.
[46] Jahnke, T., *An adaptive wavelet method for the chemical master equation*, SIAM Journal on Scientific Computing, to appear.
[47] Jahnke, T., Huisinga, W., *Solving the chemical master equation for monomolecular reaction systems analytically*, J. Math. Biol., 54 (2007) 1–26.
[48] Jensen, A., *Markoff chains as an aid in the study of Markoff processes*, Skandinavisk Aktuarietidskrift, 36 (1953) 87–91.
[49] Karypis, G., Kumar, V., *METIS: Unstructured graph partitioning and sparse matrix ordering system*, Technical Report, Computer Science Department, Univ. of Minnesota, Minneapolis, MN, 1995.
[50] Koehler, J. R., and Owen, A. B., *Computer experiments*, Handbook of Statistics, S. Ghosh and C. R. Rao (eds.), vol. 13. Elsevier Science, Oxford, UK, 261–308, 1996.
[51] Lehoucq, R. B., Sorensen, D. C., and Yang C., *ARPACK Users' Guide: Solution of Large-scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, PA, 1998.
[52] Li, H., Cao, Y., Petzold, L., and Gillespie, D., *Algorithms and software for stochastic simulation of biochemical reacting systems*, Biotech. Prog., 24 (2008) 56–61.
[53] Li, Z., Saad, Y. and Sosonkina M., *pARMS: A parallel version of the algebraic recursive multilevel solver*, Technical Report, UMSI-2001-100, Minnesota Supercomputer Institute, Univ. of Minnesota, Minneapolis, MN, 2001.
[54] Macnamara, S., Burrage, K., and Sidje, R.B., *Multiscale modeling of chemical kinetics via the master equation*, Multiscale Model. Simul., 6 (2008) 1146–1168.
[55] Mainguy, Y., Birch, J. B., and Watson, L. T., *A robust variable order facet model for image data*, Machine Vision and Applications, 8 (1995) 141–162.
[56] Micchelli, C. A., *Interpolation of scattered data: Distance matrices and conditionally positive definite functions*, Constructive Approximation Theory, 2 (1986) 11–22.
[57] Michaelis, L., Menten, M. L., *Die kinetik der invertinwirkung (Kinetics of the action of inverting)*, Biochem. Zschr., 24 (1913) 333–369.
[58] Miranker, W. L., *Numerical Methods for Stiff Equations and Singular Perturbation Problems*, D. Reidel, Dordrecht, Holland, 1981.
[59] Moler, C., Van Loan, C. F., *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev., 45 (2003) 3–49.
[60] Mouat, C.T. and Beatson, R.K., *RBF collocation*, Research Report, UCDMS 2002/3, the Mathematics and Statistics Department, Canterbury University, UK, 2002.
[61] Munsky, B., Khammash, M., *The finite state projection algorithm for the solution of the chemical master equation*, J. Chem. Phys., 124 (2006) 044104.
[62] Myers, R., *Classical and Modern Regression with Applications*, Duxbury Press, California, 2nd edition, 2000.
[63] Osio, I. G., and Amon, C. H., *An engineering design methodology with multistage Bayesian surrogates and optimal sampling*, Research in Engineering Design, 8 (1996) 189–206.
[64] Panning, T.D., Watson, L.T., Allen, N.A., Chen, K.C., Shaffer, C.A., and Tyson, J.J., *Deterministic parallel global parameter estimation for a model of the budding yeast cell cycle*, J. Global Opt., 40 (2008) 719–738.
[65] Peles, S., Munsky, B., and Khammash, M., *Reduction and solution of the chemical master equation using time-scale separation and finite state projection*, J. Chem. Phys., 125 (2006) 204104.
[66] Piegl, L., *Modifying the shape of rational B-splines. part 1:curves*, Computer-Aided Design 21, 8 (1989) 509–518.
[67] Piegl, L., *Modifying the shape of rational B-splines. part 2:surfaces*, Computer-Aided Design 21, 9 (1989) 538–546.

[68] Powell, M.J.D., *The theory of radial basis function approximation in 1990*, in Advances in Numerical Analysis vol. II, W.A. Light (eds.), Oxford University Press, Oxford, UK, 1992, 105–210.

[69] Radhakrishnan, K. and Hindmarsh, A.C., *Description and use of LSODE, the Livermore solver for ordinary differential equations*, Technical Report, LLNL report UCRL-ID-113855, Lawrence Livermore National Laboratory, Livermore, CA, USA, 1993.

[70] Rao, C. V., Arkin, A. P., *Stochastic chemical kinetics and quasi-steady-state assumption: Application to the Gillespie algorithm*, J. Chem. Phys., 118 (2003) 4999–5010.

[71] Rathinam, M., Petzold, L., Cao, Y., and Gillespie, D., *Stiffness in stochastic chemically reacting systems: the implicit tau-leaping method*, J. Chem. Phys., 119 (2003) 12784–12794.

[72] Reibman, A., Trivedi, K., *Numerical transient analysis of Markov models*, Computers and Operations Research, 15 (1988) 19–36.

[73] Renka, R. J., *Multivariate interpolation of large sets of scattered data*, ACM Transactions on Mathematical Software, 14 (1988) 139–148.

[74] Renka, R. J., *Algorithm 660: QSHEP2D: Quadratic method for bivariate interpolation of scattered data*, ACM Transactions on Mathematical Software, 14 (1988) 149–150.

[75] Renka, R. J., *Algorithm 790: CSHEP2D: Cubic Shepard method for bivariate interpolation of scattered data*, ACM Transactions on Mathematical Software, 25 (1999) 70–73.

[76] Renka, R. J., *Algorithm 791: TSHEP2D: Cosine series Shepard method for bivariate interpolation of scattered data*, ACM Transactions on Mathematical Software, 25 (1999) 74–77.

[77] Ross S. M., *Stochastic Processes*, Wiley, New York, 1983.

[78] Ross, S. M., *Approximating transition probabilities and mean occupation times in continuous-time Markov chains*, Prob. Eng. Inform. Sci., 1 (1987) 251–264.

[79] Rousseeuw, P. J., *Least median of squares regression*, Journal of American Statistical Association, 79 (1984) 871–880.

[80] Saad, Y., *SPARSKIT: a basic tool kit for sparse matrix computation, Version 2*, Technical Report, Computer Science Department, Univ. of Minnesota, Minneapolis, MN, 1994.

[81] Saad, Y., *Iterative Methods for Sparse Linear Systems, 2nd edition*, SIAM, Philadelphia, PA, 2003.

[82] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., *Design and analysis of computer experiments*, Statistical Science, 4 (1989) 409–435.

[83] Sarra, S.A. and Sturgill, D., *A random variable shape parameter strategy for radial basis function approximation methods*, Engineering Analysis with Boundary Elements, 33 (2009) 1239–1245.

[84] Schoenberg, I. J., *Metric spaces and completely monotone functions*, Annals of Mathematics, 49 (1938) 811–841.

[85] Sibson, R., and Stone, G., *Computation of thin-plate splines*, SIAM Journal on Scientific and Statistical Computing, 12 (1991) 1304–1313.

[86] Sidje, R. B., Stewart, W. J., *A survey of methods for computing large sparse matrix exponentials arising in Markov chains*, Technical Report, TR-96-06, Computer Science Department, North Carolina State University.

[87] Shepard, D., *A two-dimensional interpolation function for irregularly spaced data*, in Proceedings of the 23rd ACM National Conference, 1968, 517–523.

[88] Sidje, R. B., *Expokit: A software package for computing matrix exponentials*, ACM Trans. Math. Soft., 24 (1998) 130–156.

[89] Sjöberg, P., Lötstedt, P., and Elf, J., *Fokker Planck approximation of the master equation in molecular biology*, Computing and Visualization in Science, 12 (2009) 37–50.

[90] Stewart, W. J., *Numerical Solution of Markov Chains*, Marcel Dekker, New York, 1991.

[91] Stewart, W. J., *Introduction to the Numerical Solution of Markov Chains*, Princeton, New Jersey, 1994.

[92] Stoer, J., Bulirsch, R., *Introduction to Numerical Analysis, 2nd edition*, Springer-Verlag, New York, 1993.

[93] Sumita, U., Shanthikumar, J. G., *A software reliability model with multiple-error introduction and removal*, IEEE Trans. on Reliability, R-35 (1986) 459–462.

[94] Tyson, J. J., Novak, B., *Regulation of the eukaryotic cell cycle: molecular antagonism, hysteresis, and irreversible transitions*, J. Theo. Biol., 210 (2001) 249–263.

[95] van Kampen, N. G., *Stochastic Processes in Physics and Chemistry, 5th edition*, Elsevier, Amsterdam, 2004.

[96] van Moorsel A. P. A, Sanders, W. H., *Adaptive Uniformization*, ORSA Communications in Statistics: Stochastic Models, 10 (1994) 619–648.

[97] Versprille, K. J., *Computer-aided design applications of the rational B-spline approximation form*, Ph.D. thesis, Syracuse University, Syracuse, NY, 1975.

[98] Wang, J.G. and Liu, G.R., *On the optimal shape parameter of radial basis functions used for 2-D meshless methods*, Computer Methods in Applied Mechanics and Engineering, 191 (2002) 2611–2630.

[99] Wendland, H., *Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree*, AICM, 4 (1995) 389–396.

[100] Yoon, B. S., Shanthikumar, J. G., *Bounds and approximations for the transient behavior of continuous-time Markov chains*, Prob. Eng. Inform. Sci., 3 (1989) 175–198.

[101] Zhang, J., Watson, L., *A modified uniformization method for the chemical master equation*, in Proc. 7th IEEE Internat. Conf. on Bioinformatics and Bioengineering, Boston, MA, USA, October 2007, 1429–1433.

[102] Zhang, J.W., Watson, L.T., and Cao, Y., *Adaptive aggregation method for the chemical master equation*, Int. J. Computational Biology and Drug Design, 2 (2009) 134–148.