TENSORS AND STOCHASTIC AUTOMATA NETWORKS WITH APPLICATION TO CHEMICAL KINETICS

by

MARIE NEUBRANDER

ROGER B. SIDJE, COMMITTEE CHAIR
BRENDAN AMES
JEFF GRAY
MOJDEH RASOULZADEH
WEI ZHU

A THESIS

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

TUSCALOOSA, ALABAMA

2022

ABSTRACT

Often, given a system of biochemical reactions, it is useful to be able to predict the system's future state from the initial quantities of the involved molecules. There are many methodologies for developing such predictions, ranging from simple approaches such as Monte Carlo simulations to more sophisticated higher-order tensors and stochastic automata networks. Many revolve around solving the *chemical master equation* that arises in the modeling of the underlying biochemical kinetics. Traditionally, the chemical master equation models states consisting of the population counts present of each molecular species. This work considers the case of dealing with the resulting high dimensional data and shows how tensor representations allow us to cope with the "curse of dimensionality" that significantly complicates such problems. One key outcome in this work is the demonstration of the inherent differences and similarities between two prominent modeling methods, by computational examples on one hand and a mathematical proof on the other hand. The second key outcome is the development of a tensor-based representation of the chemical master equation modeling states as the number of times a given reaction has fired. This means that states are reaction counts as opposed to the traditional population counts. The tensor-based solutions considered in this thesis may have applications in dealing with many other high dimensional data outside of strictly chemical reaction systems.

DEDICATION

To my parents, brother, and friends for being continuous sources of support, motivation, and inspiration.

ACKNOWLEDGMENTS

I give my sincerest gratitude to Dr. Roger Sidje for serving as my advisor throughout my undergraduate career and the culmination of my work through this AMP thesis. He has taught me fundamental skills in mathematics research from doing background research, developing proofs and computational examples, and distributing my findings through writing and presentations. His investment in my work has been extremely meaningful and forms the basis that I will take with me towards my Ph.D.

I additionally thank my committee members: Dr. Jeff Gray from the UA Department of Computer Science and Dr. Brendan Ames, Dr. Mojdeh Rasoulzadeh, and Dr. Wei Zhu from the UA Deparment of Mathematics.

CONTENTS

ABSTRACT ii

DEDICATION iii

ACKNOWLEDGMENTS iv

LIST OF TABLES viii

LIST OF FIGURES x

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 BACKGROUND ON TENSORS AND STOCHASTIC AUTOMATA NETWORKS 3

2.1 Matrix and Tensor Definitions 3

2.2 Matrix Operations 4

2.3 Matrix Approximation 7

2.3.1 Singular Value Decomposition 7

2.3.2 Low Rank Approximation 8

2.3.3 Example: Image Compression 9

2.4 Tensor Operations 11

2.4.1 Rearranging Elements 11

2.4.2 Numeric Operations 12

2.5 Tensor Decomposition 13

2.5.1 Canonical Polyadic Decomposition 13
2.5.2 Tensor Train (TT) Decomposition 16
2.5.3 Quantized Tensor Train (QTT) Decomposition 17
2.6 Stochastic Automata Networks 18
2.6.1 Finite Automata and Regular Expressions 18
2.6.2 Transition Diagrams and Tables 19
2.6.3 Stochastic Automata Networks 19
CHAPTER 3 INTRODUCTION TO THE CHEMICAL MASTER EQUATION 21
3.1 General Reaction Systems and Terminology 21
3.2 Chemical Master Equation (CME) via Population Counts 25
3.2.1 Overview 25
3.2.2 Michaelis-Menten Application: Finite State Space 26
3.2.3 State Space Truncation 29
CHAPTER 4 CHEMICAL MASTER EQUATION VIA TENSORS 32
4.1 Classic Approach 32
4.1.1 Algorithm 33
4.1.2 Application Example: Michaelis-Menten 36
4.2 SANs Approach 37
4.2.1 Notations and Modeling Method 37
4.2.2 Algebraic Version and Algorithm 39
4.2.3 Application Example: Michaelis-Menten 42
4.3 Methodology Comparison 42

CHAPTER 5 CHEMICAL MASTER EQUATION VIA REACTION COUNTS 50

5.1 Set-Up 50

5.2 Solving the CME via Reaction Counts 51

5.3 Tensor-Based Reaction Count CME Operator 53

5.4 Computational Example 55

5.5 Adaptive Solution to the Reaction Count CME 57

CHAPTER 6 CONCLUSIONS 59

REFERENCES 61

LIST OF TABLES

2.1 Summary of storage complexities of explicit, TT, and QTT tensor forms. 18

3.1 Michaelis-Menten Reaction System. 23

5.1 Growth Decay Reaction System. 51

5.2 Computational results for CME probability leaks via reaction counts with $x_0 = [3, 2, 2, 2]^T$. A value of $\sum q(t)$ closer to 1 indicates less probability leak. 57

LIST OF FIGURES

2.1 Visualization of 1^{st}, 2^{nd}, and 3^{rd} order tensors in R^I, $\mathsf{R}^{I\ J}$, and $\mathsf{R}^{I\ J\ K}$, respectively. 4

2.2 1D, 2D, and 3D gridpoint visualization. 4

2.3 Low rank approximations of a black and white image. 10

2.4 Visual representation of CP Decomposition on a third order tensors; from [1] 14

2.5 First four frames of a MATLAB movie before (left) and after (right) CP Decomposition 16

2.6 Visual representation of Tensor Train Decomposition. 17

2.7 DFA transition diagram (left) and table (right) for the regular expression $(ajb)^{\star}abb$. 19

3.1 Michaelis-Menten Reaction System. 24

3.2 Michaelis-Menten 5-State Stochastic Automata Example 27

3.3 Probabilities of being in states $\mathsf{x}_1 \ldots \mathsf{x}_5$ at times 0, 20, 40, 60, 80, and 100. 29

3.4 Marginal probabilities of each species' quantity at times t =0, 20, 40, 60, 80, 100. Each row corresponds to a time. Each column corresponds to a molecule. x-axis indicates population counts; y-axis gives corresponding probabilities. 31

4.1 The shape and 42 non-zero values of the CME operator generated in Section 4.1.2. Columns sum to zero, except on boundary states where probability leaks occur. 37

4.2 The shape and 32 non-zero values of the CME operator from Section 4.2.3. Rows sum to zero. 42

5.1 Example shapes of population count and reaction count CME operators modeling the same system. 56

5.2 Marginal population count probabilities computed via population and reaction counts for Michaelis-Menten at t = 0.5. Population count marginals in 5.2b are recovered from the distribution computed by the reaction count method. 58

CHAPTER 1

INTRODUCTION

There is an ever-increasing need to more effectively handle large quantities of data in many important fields of study. This data is generally *high dimensional*, meaning it captures a large number of variables, and it must be stored in a way that allows for efficient analysis and computations. My work uses *higher-order tensors* for improved space and computational efficiency. An order-N tensor is an N-dimensional array that can be in the shape of a vector ($N = 1$), matrix ($N = 2$), or a higher-order tensor ($N \quad 3$). While these data structures are, inherently, storage-intensive, there exist many decompositions and compressions that allow for large efficiency improvements, making higher-order tensors an appealing method of coping with large quantities of data. My work focuses on the applications of these higher order tensors specifically with regards to modeling biochemical reaction systems via the *chemical master equation* (CME).

Such reaction systems are found in a wide array of biological settings including enzyme kinetics and genetics. Additionally, these reaction systems can be useful for modeling problems beyond the scope of molecular chemical reactions. For instance, one method of modeling disease spread is to view the transmission and recovery processes as a set of chemical reactions: one infected individual and one health individual may "react" to produce two infected individuals. Analyzing and developing efficient higher-order tensor representations of this problem promises to be impactful both in and beyond the scope of biochemical reaction systems.

The material in this Master's thesis builds on a faculty-reviewed paper that appeared in SIURO (SIAM Undergraduate Research Online Journal) [2]. An outline of the document is as follows:

- Chapter 2 introduces background regarding matrices, tensors, and stochastic automata networks.
- Chapter 3 introduces the fundamentals of modeling biochemical reaction systems via the chemical master equation. In this chapter, the states of the chemical master equation are formulated as population counts of each molecular species involved in the system.
- Chapter 4 introduces and analyzes similarities/differences between two separately-published tensor-based formulations of the chemical master equation from Chapter 3.
- Chapter 5 reformulates the chemical master equation in terms of reaction counts. In this chapter, states represent the number of times each reaction has fired. This chapter introduces a new tensor-based formulation of the CME.

CHAPTER 2

BACKGROUND ON TENSORS AND STOCHASTIC AUTOMATA NETWORKS

This chapter introduces necessary terminology and operations for matrices and tensors. Included are three decomposition/approximation methods: Singular Value Decomposition, Canonical Polyadic (CP) Decomposition, and Tensor Train (TT) Decomposition.

2.1 Matrix and Tensor Definitions

An N-dimensional (or order-N) tensor is an array $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ where for $n = 1, 2, \ldots, N$, the size of the n^{th} dimension, also known as the n^{th} mode, is I_n and the indexing range is therefore 1 to I_n (or 0 to $I_n - 1$ if indexing is zero-based); elements are referenced by the notation $\mathsf{X}_{i_1,i_2,\ldots,i_N} = \mathsf{X}(i_1, i_2, \ldots, i_N)$ for i_n in the n^{th} dimension's indexing range [1]. Additional referencing notations are:

- A colon subscript indicates all elements of a mode.
- A *fiber* is given by fixing all indices of the tensor except for one. A mode-n tensor fiber is given by fixing the indices of all modes except for the n^{th}.
- A *slice* is given by fixing all indices of the tensor except for two.

As shown in Figure 2.1, one-dimensional and two-dimensional tensors are vectors and matrices, respectively; a three-dimensional tensor may be visualized in a 3D structure and thought of as several equal-sized matrices stacked on top of each other.

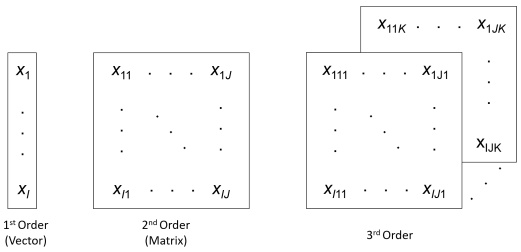


Figure 2.1: Visualization of 1^{st}, 2^{nd}, and 3^{rd} order tensors in $\mathbb{R}^I$, $\mathbb{R}^{I\ J}$, and $\mathbb{R}^{I\ J\ K}$, respectively.

While these can be easily visualized in 1D, 2D, and 3D as gridpoints as is shown in Figure 2.2, visualization is harder in higher dimensions, but the concept of gridpoints extends to integer lattices or hyper-rectangles.

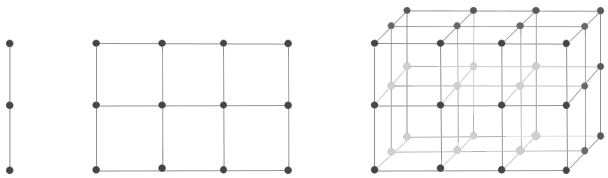


Figure 2.2: 1D, 2D, and 3D gridpoint visualization.

2.2 Matrix Operations

With the notations used to represent tensors introduced, we now delve into operations used for tensor-based computations, starting with the simplest case: matrix (and vector) operations. Matrix addition and multiplication are widely known standard operations; in addition to these, we introduce several more advanced and less widely known operations [1]. Extensions of these operations to higher order tensors are discussed in Section 2.4.

Addition and Multiplication

Matrix addition is performed on two $I \quad J$ matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ where the resultant matrix $\boldsymbol{C}$ has element c_{ij} given by

$$c_{ij} = a_{ij} + b_{ij}$$

Standard matrix multiplication between an $I \times J$ matrix $\mathbf{A}$ and an $J \times K$ matrix $\mathbf{B}$ yields an $I \times K$ matrix $\mathbf{C}$ with element c_{ik} given by

$$c_{ik} = \sum_{j=1}^{J} a_{ij} b_{jk}.$$

Diagonal Creation

For a vector $\mathbf{a} \in \mathbb{R}^I$, the *diagonalization* of $\mathbf{a}$ is denoted by $\text{diag}(\mathbf{a})$ and results in a square $I \times I$ matrix with elements

$$\text{diag}(\mathbf{a}) = \begin{bmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_I \end{bmatrix}.$$

Frobenius Norm

Often in dealing with matrices/tensors, it is essential to have a measure of how different or similar two equally-sized objects are. One method of doing so is the *Frobenious norm*. The Frobenius norm of an $m \times n$ matrix A is given by

$$\|A\| = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}.$$

Kronecker Product

The Kronecker Product of two matrices $\boldsymbol{A}$ (size $I \quad J$) and $\boldsymbol{B}$ (size $K \quad L$) is denoted $\boldsymbol{A} \quad \boldsymbol{B}$ and results in a matrix of size $(IK) \quad (JL)$. It is defined as

$$\boldsymbol{A} \quad \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \dots & a_{1J}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \dots & a_{2J}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\boldsymbol{B} & a_{I2}\boldsymbol{B} & \dots & a_{IJ}\boldsymbol{B} \end{bmatrix}.$$

Kronecker Sum

The Kronecker Sum [3] of an $I \quad I$ square matrix $\boldsymbol{A}$ and a $J \quad J$ square matrix $\boldsymbol{B}$ is given by

$$\boldsymbol{A} \quad \boldsymbol{B} = \boldsymbol{A} \quad \boldsymbol{II}^{J} + \boldsymbol{II}^{I} \quad \boldsymbol{B}$$

where $\boldsymbol{II}^{K}$ is the $K \quad K$ identity matrix. For example, the Kronecker sum of two $2 \quad 2$ matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ is given by

$$\boldsymbol{A} \quad \boldsymbol{B} = \begin{bmatrix} a_{11} + b_{11} & b_{12} & a_{12} & 0 \\ b_{21} & a_{11} + b_{22} & 0 & a_{12} \\ a_{21} & 0 & a_{22} + b_{11} & b_{12} \\ 0 & a_{21} & b_{21} & a_{22} + b_{22} \end{bmatrix}.$$

For diagonal matrices, the result is simpler: the Kronecker sum of two diagonal matrix is another diagonal matrix. Take diagonal matrices $\boldsymbol{A} \; 2 \; \mathrm{R}^{I \quad I}$ and $\boldsymbol{B} \; 2 \; \mathrm{R}^{J \quad J}$ with $\boldsymbol{A} = \mathrm{diag}(\boldsymbol{a}), \boldsymbol{a} \; 2 \; \mathrm{R}^{I}$ and $\boldsymbol{B} = \mathrm{diag}(\boldsymbol{b}), \boldsymbol{b} \; 2 \; \mathrm{R}^{J}$. Then

$$\boldsymbol{A} \quad \boldsymbol{B} = \mathrm{diag}([a_1 + \boldsymbol{b}, a_2 + \boldsymbol{b}, \dots a_I + \boldsymbol{b}])$$

where a_i + $\boldsymbol{b}$ is defined by $[a_i + b_1, \ldots, a_i + b_J]$. The vector $[a_1 + \boldsymbol{b}, a_2 + \boldsymbol{b}, \ldots a_I + \boldsymbol{b}]$ may be generated by

$$[a_1 + \boldsymbol{b}, a_2 + \boldsymbol{b}, \ldots a_I + \boldsymbol{b}] = \boldsymbol{a} \quad \mathbb{1}^J + \mathbb{1}^I \quad \boldsymbol{b} \tag{2.1}$$

where $\mathbb{1}^k$ is the vector $[1, \ldots, 1]$ 2 $\mathbb{N}^k$. During this work, we may refer to the "Kronecker sum" of two vectors as

$$\boldsymbol{a} \quad \boldsymbol{b} = \boldsymbol{a} \quad \mathbb{1}^J + \mathbb{1}^I \quad \boldsymbol{b}$$

noting that this is a slight abuse of notation.

Khatri-Rao Product

The Khatri-Rao Product of two matrices $\boldsymbol{A}$ (size $I \quad J$) and $\boldsymbol{B}$ (size $K \quad J$) is denoted $\boldsymbol{A} \quad \boldsymbol{B}$ and results in a matrix of size $(IK) \quad (J)$. It is defined as

$$\boldsymbol{A} \quad \boldsymbol{B} = [\boldsymbol{a}_1 \quad \boldsymbol{b}_1 \; \boldsymbol{a}_2 \quad \boldsymbol{b}_2 \; \ldots \; \boldsymbol{a}_J \quad \boldsymbol{b}_J]$$

2.3 Matrix Approximation

While matrices (and higher order tensors) are extremely useful for representing data, they can become extremely difficult to deal with in terms of storage and computation time. Matrix approximations are methods of representing matrices that use less space than would be required for storing each element of the matrix; here, we present the *low rank approximation* that makes use of a matrix's *singular value decomposition*.

2.3.1 Singular Value Decomposition

Given a matrix, we can perform a *decomposition* in which we write the matrix as a product of several others. Here, we will investigate one such method known as

singular value decomposition. Given some $m \quad n$ matrix $\boldsymbol{A}$, we can perform a ***singular value decomposition***, writing $\boldsymbol{A}$ in the form

$$\boldsymbol{A} = \boldsymbol{U} \quad \boldsymbol{V}^T$$

Here, $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal matrices and $\quad$ is a diagonal matrix. $\boldsymbol{U}$ is $m \quad n$ and contains the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$, $\boldsymbol{V}$ is $n \quad n$ and contains the eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$, and $\quad$ is $n \quad n$ and contains the singular values of $\boldsymbol{A}$ (the i^{th} singular value is σ_i).

Letting r be the rank of $\boldsymbol{A}$, σ_i be the i^{th} diagonal element of S (the i^{th} largest singular value), $\boldsymbol{u}_i$ be the i^{th} column of $\boldsymbol{U}$, and $\boldsymbol{v}_i$ be the i^{th} column of $\boldsymbol{V}$, we can write the SVD as

$$\boldsymbol{A} = \boldsymbol{U} \quad \boldsymbol{V} = \sum_{i=1}^{r} \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i^T$$

Notice that each operation $bmu_i\sigma_i\boldsymbol{v}_i^T$ yields a row vector times a column vector, which gives a matrix - this type of vector product is referred to as an ***outer product***, and will be important for later understanding of higher-order tensor decomposition.

2.3.2 Low Rank Approximation

In singular value decomposition, we are writing a representation of the entire matrix - we can build from these results using ***Low Rank Approximation*** to write an approximation of the original matrix using fewer rows and columns. To find the best rank-l, $l < r$ approximation $\boldsymbol{A}^{\emptyset}$ for the original matrix $\boldsymbol{A}$ using low-rank approximation, we perform

$$\boldsymbol{A}^{\emptyset} = \sum_{i=1}^{l} u_i \sigma_i v_i^T$$

Alternatively, we can write this approximation using a matrix $\quad^{\emptyset}$, where the smallest $r \quad l$ singular values of $\quad$ are replaced with zeroes. Using this method, we write

$$\boldsymbol{A}^{\emptyset} = \boldsymbol{U} \quad ^{\emptyset}\boldsymbol{V}^T = \boldsymbol{U}\,\mathrm{diag}(\sigma_1\sigma_2...\sigma_l, 0, 0...)\boldsymbol{V}^T$$

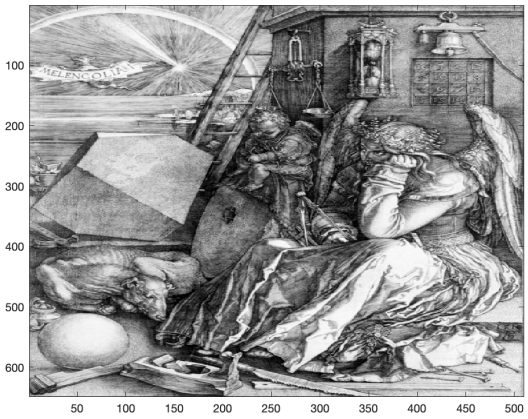
One question that arises from approximating matrices is *how good was the approximation?*; one way of analyzing this is using the Frobenius norm of the difference between the original matrix and the approximation, $kA \quad A'k$. For a perfect approximation ($A = A'$), this result would be zero; the larger the value, the worse the approximation. To illustrate the savings of a low-rank approximation, assume $n = m$ and $r \quad n$. Then, while the full matrix would take a quadratic $O(n^2)$ storage, the low-rank approximation would take a linear $O(nr)$.

2.3.3 Example: Image Compression

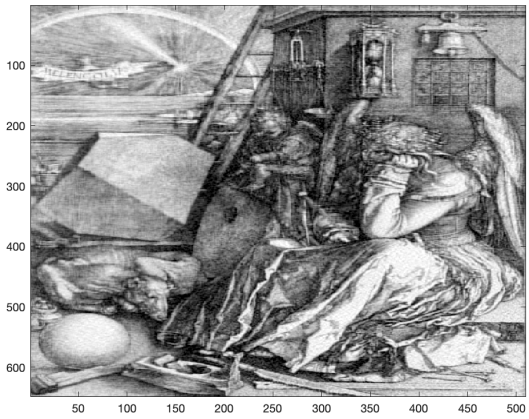
Low rank approximation for singular value decomposition is a useful tool as it reduces the amount of data needed to represent a matrix; one illustrative application field is in image compression.

Images (particularly black and white images) are naturally represented as matrices. Given a black and white image with height h pixels and length l pixels, the corresponding matrix is $A \; 2 \; R^{h \;\; l}$. Entry a_{ij} is a value representing how dark or light the pixel at row i column j is. For large images, this can require lots of storage; a low rank approximation A' can be used to represent A using less data - of course, the lower the rank, the more blurry the image will become.

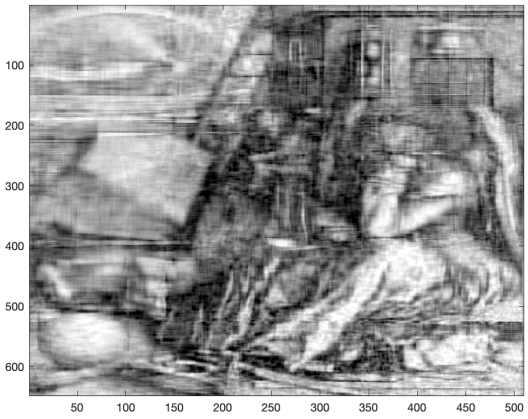
Figure 2.3 shows an example of this. Image 2.3a is the original 648 500 image. Images 2.3b, 2.3c, and 2.3d show rank 100, 30, and 10 approximations, respectively. Note that the rank 100 approximation is nearly indistinguishable from the original image. The image is still distinguishable in the rank 30 approximation, but many details are lost. By the rank 10 approximation, the general form of the image can be found but is largely indistinguishable on its own.



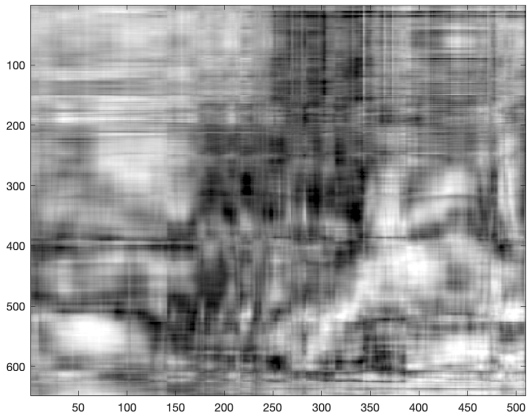
(a) Original image.



(b) Rank 100 compression.



(c) Rank 30 compression.



(d) Rank 10 approximation.

Figure 2.3: Low rank approximations of a black and white image.

2.4 Tensor Operations

Now that matrix operations and approximations have been introduced, we move to operations for higher order tensors [1]. First, we introduce methods for rearranging tensor elements; from there, we introduce operations analagous to matrix operations.

2.4.1 Rearranging Elements

One category of operation that can be done to a tensor is element rearrangement; the two introduced here are *vectorization* and *matricization*. These allow for easier viewing and handling of higher-order tensors.

Vectorization

The elements of a tensor can be represented in vector form vec(X) in which fibers are connected in a single line in a chosen, well-defined order. The following is an example of a vectorization of a mode-3 tensor:

$$\text{vec}(\mathcal{X}) = [x_{111}, x_{112}, ...x_{211}, x_{212}...x_{311}, x_{312}, ...x_{333}]^T$$

Matricization

Similar to vectorization, matricization is the process of representing a tensor in the format of a matrix. A mode-n matricization is the representation in which the mode-n fibers of the tensor are the columns of the matrix. The mapping of tensor element $(i_1, i_2, ...i_N)$ to matrix element (i_n, j) is given by

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1) J_k \text{ with } J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m$$

2.4.2 Numeric Operations

Addition and Subtraction

Addition and subtraction extend nicely from matrices to higher-order tensors - these operations are performed element-wise. Given two tensors $\mathcal{X}$ and $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the elements of their sum $\mathcal{Z}$ are given by

$$\mathcal{Z}_{i_1,i_2,\ldots i_N} = \mathcal{X}_{i_1,i_2,\ldots i_N} + \mathcal{Y}_{i_1,i_2,\ldots i_N}$$

Multiplication

Unlike addition, multiplication does not have a clear extension to higher-order tensors. However, one extension of multiplication to tensors is the *n-mode matrix product* between a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $U \in \mathbb{R}^{J \times I_n}$, notated $\mathcal{X} \times_n U$. The resulting product is a tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$, and the elements of $\mathcal{Y} = \mathcal{X} \times_n U$ are given by

$$\mathcal{Y}_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N} u_{j i_n}$$

Alternatively, this can be represented through mode-n matricized tensors, where

$$Y_{(n)} = U X_{(n)}$$

Other complex ways of defining tensor multiplication exist [1].

Norm

The *norm* of a tensor is analogous to the Frobenius norm of a matrix, and is given by

$$\|\mathcal{X}\| = \sqrt{\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\cdots\sum_{i_n=1}^{I_N} x_{i_1\ldots i_N}^2}$$

.

2.5 Tensor Decomposition

Storing higher-order tensors can consume large amounts of data. Just as there are approximation techniques for matrices, there are also many existing ways for approximating higher-order tensors.

2.5.1 Canonical Polyadic Decomposition

One form of tensor decomposition is Canonical Polyadic (CP) Decomposition; this is very similar to singular value decomposition for matrices as was presented in Section 2.3.1. Recall that the singular value decomposition of a matrix is written as the sum of several outer products of vectors. The concept of an outer product (notated $\circ$) can be extended into higher dimensions with the outer product of N vectors yielding a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ as follows:

$$\mathcal{X} = \mathbf{a}_1 \circ \mathbf{a}_2 \circ \ldots \circ \mathbf{a}_N, \quad \text{where} \quad x_{i_1 i_2 \ldots i_N} = a_{1_{i_1}} a_{2_{i_2}} \ldots a_{N_{i_N}}$$

The application of the outer product to CP decomposition can be nicely illustrated with a third order tensor. Using CP Decomposition, a third order tensor is represented

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

where $\mathbf{a}_r$, $\mathbf{b}_r$, and $\mathbf{c}_r$ represent vectors, the products $\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ are third-order tensors, and R represents the rank of the resulting tensor.

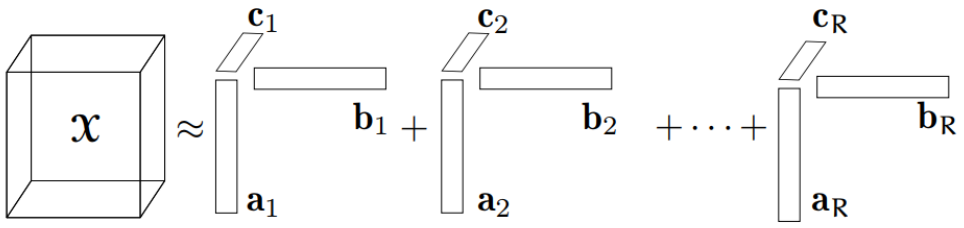


Figure 2.4: Visual representation of CP Decomposition on a third order tensors; from [1]

This results of a CP Decomposition can be written as factor matrices, each containing the vectors corresponding to a single dimension. For a third order tensor, we have three factor matrices: $A = [a_1\ a_2\ ...\ a_R]$, $B = [b_1\ b_2\ ...\ b_R]$, and $C = [c_1\ c_2\ ...\ c_R]$. Employing these allows us to write X in matricized form as follows:

$$X_{(1)} \quad A(C \quad B)^T$$

$$X_{(2)} \quad B(C \quad A)^T$$

$$X_{(3)} \quad C(B \quad A)^T$$

When decomposing a tensor with CP decomposition, there will likely be some approximation error - given an initial tensor X, constructing the tensor given by the factor matrices will give a different tensor X'. The error between these can be given by

$$error : \|\mathsf{X} \quad \mathsf{X}'\|$$

The size of the error is dependent on the rank of the resulting approximation; picking a value for the rank is a complex problem and is not further explored here.

Algorithms

When dealing with CP decomposition, two of the most important algorithms are performing the decomposition to attain factor matrices and reconstructing a tensor

given factor matrices. The former is performed by fixing all but the first factor matrix, solving for it; fixing all but the second factor matrix and solving for it; and so on, repeating until a stopping condition, such as a low enough error, is met. The latter is performed through summing the outer products of corresponding columns of factor matrices. Very simple examples of algorithms for a third order tensor are shown below.

Algorithm 2.5.1 CP Decomposition

Data: X

A, B, C randomly initialized

while *stopping condition not met* do

$A^T \quad (C^T C \quad B^T B)^{\ 1}(X_{(1)}(C \quad B))^T$

$B^T \quad (C^T C \quad A^T A)^{\ 1}(X_{(2)}(C \quad A))^T$

$C^T \quad (B^T B \quad A^T A)^{\ 1}(X_{(3)}(B \quad A))^T$

X from Algorithm 2.5.2

end

Algorithm 2.5.2 Tensor Construction

Data: A, B, C

initialize X to all zeros

while *i less than Rank* do

$tempX = a_i \quad b_i \quad c_i$

X = X + *tempX*

end

Example: Movie Frames

One visual example of decomposing a tensor is decomposing a movie, which consists of several frames stacked on top of each other. In MATLAB, a “movie" of several plots playing one after another can be created and stored as a three-dimensional tensor such that each slice represents a frame/plot. CP Decomposition can then be performed on the tensor and the movie frames can be played such that less storage is required but the contents of the frames are retained. Figure 2.5 below shows the first few frames of such a “movie" before and after a CP Decomposition. Of important note, the CP Decomposition performed here is not using my CPD function, but rather a more sophisticated decomposition from the tensorlab package.

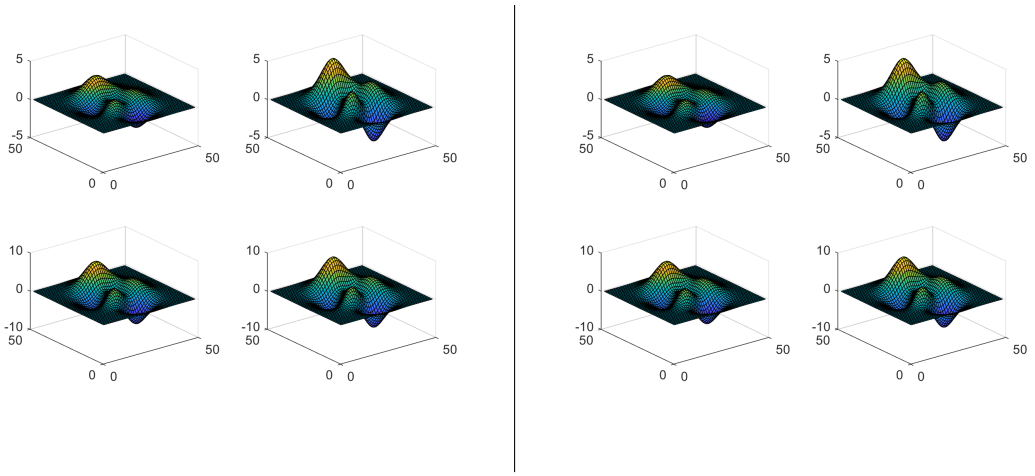


Figure 2.5: First four frames of a MATLAB movie before (left) and after (right) CP Decomposition

2.5.2 Tensor Train (TT) Decomposition

In application to the chemical master equation, CP decomposition is not used in practice; more commonly, we see what is known as Tensor Train Decomposition [4]. Below is a brief background on tensor train to lay the groundwork towards investigating that problem; notations are similar to those of [5, 6].

In the tensor train (TT) decomposition of a tensor $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, one is able to compute the value of $\mathsf{X}_{i_1,i_2,\ldots,i_N}$ through a series of vector and matrix multiplications derived from a new set of N 3-dimensional tensors $\mathsf{X}_1, \ldots, \mathsf{X}_N$ referred to as the TT cores of X. For $i_k \in \{1, \ldots, I_k\}$, $1 \leq k \leq N$, the i_k^{th} slice of tensor X_k is written $\mathsf{X}_k(i_k)$. Using this notation, the TT decomposition of X is written as

$$\mathsf{X}(i_1, i_2, \ldots, i_N) = \mathsf{X}_1(i_1) \cdots \mathsf{X}_N(i_N)$$

Matrix dimensions are given by $\mathsf{X}_k(i_k) \in \mathbb{R}^{r_{k-1} \times r_k}$ with boundary conditions $r_0 = r_N = 1$; these values are referred to as the TT ranks of X. The dimensions of the TT cores are $\mathsf{X} \in \mathbb{R}^{r_{k-1} \times I_k \times r_k}$.

Letting $I := \max(I_1, \ldots, I_N)$ and $r_{TT} := \max(r_1 \ldots r_{N-1})$, the storage cost of the TT decomposition is $O(r_{TT}^2 N I)$; when r_{TT} is small, this is substantially less than the explicit storage $O(I^N)$. For such cases, the TT decomposition is promising in chemical master equation applications, where tensor representations can become very large.

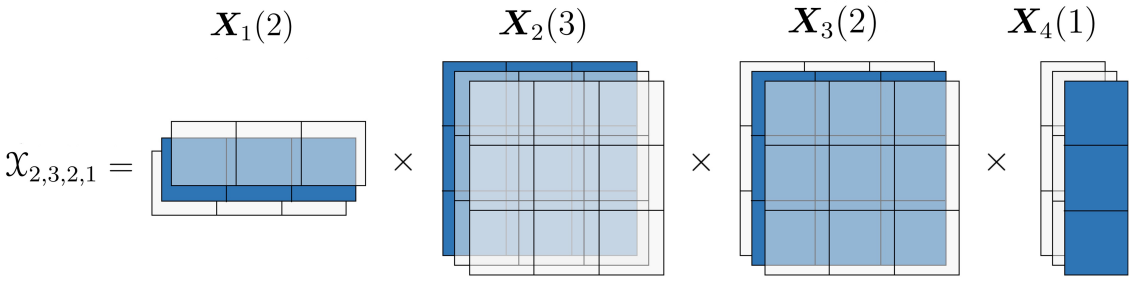


Figure 2.6: Visual representation of Tensor Train Decomposition.

In Figure 2.6, a 4-dimensional tensor $\mathbf{X}$ has been decomposed into four 3-dimensional tensors, $\mathbf{X_1}, \mathbf{X_2}, \mathbf{X_3}$, and $\mathbf{X_4}$. The value at $\mathbf{X}_{2,3,2,1}$ is calculated by multiplying the row-vector from the 2^{nd} frontal slice of $\mathbf{X_1}$, the matrix from the 3^{rd} frontal slice of $\mathbf{X_2}$, the matrix from the 2^{nd} frontal slice of $\mathbf{X_3}$, and the column vector from the 1^{st} frontal slice of $\mathbf{X_4}$.

2.5.3 Quantized Tensor Train (QTT) Decomposition

An alternative form of the TT decomposition providing additional storage benefitss is the Quantized Tensor Train (QTT) decomposition; the QTT decomposition is popular in the area of modeling biochemical reaction systems largely due to the work of [7]. Via the QTT decomposition, a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \cdots I_N}$ with $I_s = 2^{L_s}$ for some $L_s \in \mathbb{N}$ can be reshaped into a tensor $\mathbf{Y}$ of order $L = L_1 + \ldots + L_N$ and size $2 \times \ldots \times 2$ (L times). Take the binary representation of $i_s - 1$ given by

$$i_s - 1 = \left(\ell_1^{(s)} - 1\right) + \ldots + 2^{L_s - 1}\left(\ell_{L_s}^{(s)} - 1\right)$$

where $\ell_k^{(s)}$ 1 2 f0, 1g. Then the QTT yields

$$\mathsf{Y}\left(\ell_1^{(1)},\ldots,\ell_{L_1}^{(1)},\ell_1^{(2)},\ldots\ell_{L_2}^{(2)},\ldots,\ell_1^{(N)}\ldots\ell_{L_N}^{(N)}\right) = \mathsf{X}(i_1,i_2,\ldots,i_N)$$

Placing tensor Y into TT format gives the QTT form of X. Let r_{QTT} be the maximum of the TT ranks of Y. The storage complexity of the QTT form of X is then $O(r_{QTT}^2 \quad N \quad \log_2(I))$, an even larger reduction from explicit storage cost than the TT. These storage complexities are summarized in Table 2.1

Format	Explicit	TT	QTT
Storage	I^N	$O(r_{TT}^2 \quad N \quad I)$	$O(r_{QTT}^2 \quad N \quad \log_2(I))$

Table 2.1: Summary of storage complexities of explicit, TT, and QTT tensor forms.

2.6 Stochastic Automata Networks

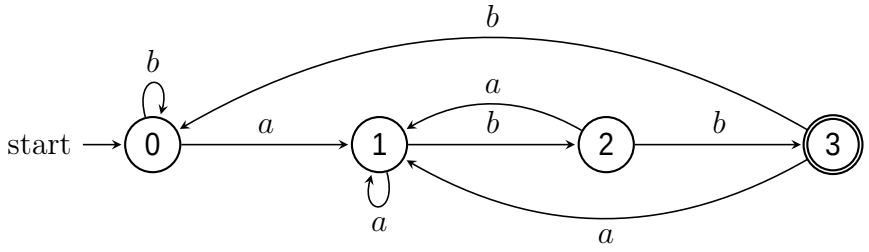
Now that we have introduced tensors, we turn to *automata networks* with an underlying continuous time Markov Chain methodology.

2.6.1 Finite Automata and Regular Expressions

Before expanding to the concept of stochastic automata, we will first introduce *finite automata*, a simple but powerful mechanism for machines to recognize patterns. Given an input, a finite state automata can transition between various states and determine whether the final state matches a desired output [8]. There exist two types of finite automata: non-deterministic (NFA) and deterministic (DFA); our focus will be on DFA. DFA consist of a set of states (including a starting state and a final state), set of inputs, and a function indicating how states transition between each other. Additionally, in DFA, any given set of inputs leads to exactly one set of transitions and one output [8]. One useful way of representing DFA is via diagrams or tables of state changes.

2.6.2 Transition Diagrams and Tables

The concept of DFA becomes substantially clearer through an example. For instance, say we wanted to represent the regular expression $(ajb)^\star abb$ using DFA (as is presented in [8]). Here, we are taking in strings of any length of $a^\emptyset s$ and $b^\emptyset s$; we are looking specifically for a string beginning with any combinations of a and b, followed by the sequence abb. In this example, a and b make up the input alphabet and the state of reaching abb is the final state. Figure 2.7 contains the corresponding DFA transition diagram and table, where states have been enumerated from 0 to 3 with 0 as the starting state and 3 as the ending state. In the diagram, the arrows indicate the state transitions caused by the set of inputs. In the table, the left “State" column indicates a state before a transition; after input a or b, there is a transition to the state in the corresponding column. A dash indicates there is no change and corresponds to a self-loop in the diagram.



State	a	b
0	1	-
1	-	2
2	1	3
3	1	0

Figure 2.7: DFA transition diagram (left) and table (right) for the regular expression $(ajb)^\star abb$.

2.6.3 Stochastic Automata Networks

In order to outline *stochastic automata networks*, we must first define *stochastic automata*. Stochastic automata with finite number of states are similar to finite automata in that there is a finite set of states with a transition function mapping a next state for each given state; like DFA, a single transition from a particular state leads to a unique state. However, unlike in finite automata, in stochastic automata,

each transition has a certain probability of happening. For a given state, the sum of probabilities of leaving the state via different transitions should sum to one. A stochastic automata network is then formed by analyzing the interaction of several stochastic automata together; an example of this in the context of chemical reaction systems will be given in Section 3.2.2.

CHAPTER 3

INTRODUCTION TO THE CHEMICAL MASTER EQUATION

The concepts introduced in the preceding sections can be used to model systems of biochemial reactions via the *chemical master equation* (CME). Section 3.1 introduces important notations and concepts used in reaction systems and the CME; Section 3.2 uses these terminologies in demonstrating the CME.

3.1 General Reaction Systems and Terminology

When modelling a system of reactions, there are several key components, including the species of molecules involved in the reactions, the quantities of each species present, the equations for the reactions themselves, and the rates at which the reactions occur; the notations presented in this section are similar to but modified from notations seen in other literature in the field, such as in [7, 9]. For a given reaction system, define N to be the number of different types of molecular species involved; the species are labeled as $S_1 \dots S_N$. To describe the quantities of the various species present at some point in time, let $\mathbb{N}$ be the set of natural numbers including 0. Define a *population count state* to be a nonnegative integer vector

$$\boldsymbol{x} := (x_1, x_2, \dots, x_N)^T \in \mathbb{N}^N,$$

where $\boldsymbol{x}$ is the state with quantity x_l of species S_l. For a system with j j possible states, the *state space* is defined as the set

$$= \left\{ \boldsymbol{x}_i = (x_1^{(i)}, x_2^{(i)}, \dots, x_N^{(i)})^T \in \mathbb{N}^N \right\}_{i=1}^{j \; j}. \tag{3.1}$$

We will use $\dot{\imath}(\boldsymbol{x})$ to reference the index of the state $\boldsymbol{x}_i$ in the enumeration (3.1). Transitions between states occur via a number of M chemical reactions R_k for $k = 1, 2 \ldots, M$. The equations for the reactions are given by

$$R_k : s_{k1}S_1 + s_{k2}S_2 + \ldots + s_{kN}S_N \xrightarrow{c_k} s_{k,N+1}S_1 + s_{k,N+2}S_2 + \ldots + s_{k,2N}S_N \tag{3.2}$$

where the positive $c_k \in \mathbb{R}$ is the *rate constant* of reaction $R_i k$. The values $s_{k1}, \ldots s_{k,2N}$ are *stoichiometric coefficients* notating how much of each species is involved in either side of the reaction. The coefficient of species S_l on the left-hand side of reaction R_k is given by s_{kl}; its coefficient on the right-hand side is given by $s_{k,N+l}$. A convenient way of representing this information is via an $M \times 2N$ *stoichiometry matrix* of the form

$$\left[\begin{array}{ccc|ccc} s_{11} & \cdots & s_{1N} & s_{1,N+1} & \cdots & s_{1,2N} \\ \vdots & & \vdots & \vdots & & \vdots \\ s_{M1} & \cdots & s_{MN} & s_{M,N+1} & \cdots & s_{M,2N} \end{array}\right] \in \mathbb{N}^{M \times 2N}. \tag{3.3}$$

where entry (k, l) represents the corresponding coefficient s_{kl}. Using a similar notation, we also define an $N \times M$ *stoichiometry change matrix* $\boldsymbol{V}$ that indicates how much each reaction changes the quantities of each species. This matrix is written in transposed form for consistency of notation as

$$\boldsymbol{V} = \begin{bmatrix} s_{1,N+1} & s_{11} & \cdots & s_{1,2N} & s_{1N} \\ \vdots & & \ddots & \vdots & \\ s_{M,J+1} & s_{I1} & \cdots & s_{M,2N} & s_{MN} \end{bmatrix}^T \in \mathbb{Z}^{N \times M}. \tag{3.4}$$

In the stoichiometry change matrix, the k^{th} reaction is associated with the k^{th} column of $\boldsymbol{V}$ given by $\boldsymbol{v_k} = (v_{1k}, \ldots, v_{N_k})^T := (s_{k,N+1} \quad s_{k,1} \ldots s_{k,2N} \quad s_{k,N})^T$. Element v_{lk} gives the quantity by which reaction R_k changes the population count of species S_l. When the system is in state $\boldsymbol{x}$, the firing of reaction R_k causes the transition to state $\boldsymbol{x} + \boldsymbol{v_k}$.

Associated with each reaction is a state-dependent *propensity function* $\alpha_k(\mathbf{x})$. Usually, this function is given by

$$\alpha_k(\mathbf{x}) = c_k \prod_{l=1}^{N} \binom{x_l}{v_{lk}} \tag{3.5}$$

assuming that for any $x_l < v_{lk}$, $\binom{x_l}{v_{lk}} = 0$ and for any $v_{lk} = 0$, $\binom{x_j}{v_{lk}} = 1$. This combinatorial components indicates how many ways there are to select the v_{lk} instances of species S_l from the x_l present in state $\mathbf{x}$ in order for reaction R_k to fire. It will become important to note that in this case, $\alpha_k(\mathbf{x})$ is *separable*: there exist functions $\alpha_k^{(1)} \dots \alpha_k^{(N)}$ such that for $\mathbf{x} = (x_1, \dots, x_N)^T$, the propensity function can be derived by $\alpha_k(\mathbf{x}) = \prod_{l=1}^{N} \alpha_k^{(l)}(x_l)$.

Michaelis-Menten Reaction System

The Michaelis-Menten reaction system is a well-known approach to modeling enzyme kinetics and is based around the interactions of substrates and enzymes. In Michaelis-Menten kinetics, a substrates (S) will bind with an enzyme (E) to form an enzyme-substrate complex (ES). The enzyme-substrate complex can then either reverse back into the substrate and enzyme or react irreversibly to yield a product (P) and the original enzyme; this process is illustrated in Figure 3.1.

Throughout this exposition, the Michaelis-Menten species S_1, S_2, S_3, S_4 are represented by S, E, ES, P, respectively. For instance, the state $[2, 1, 0, 0]$ indicates the presence of two substrate S, one enzyme E, zero intermediate ES, and zero product P. The three reactions modeled by this system are defined in Table 3.1

$$R_1 : S + E \xrightarrow{c_1} ES$$
$$R_2 : S + E \xrightarrow{c_2} ES$$
$$R_3 : ES \xrightarrow{c_3} P + E$$

Table 3.1: Michaelis-Menten Reaction System.

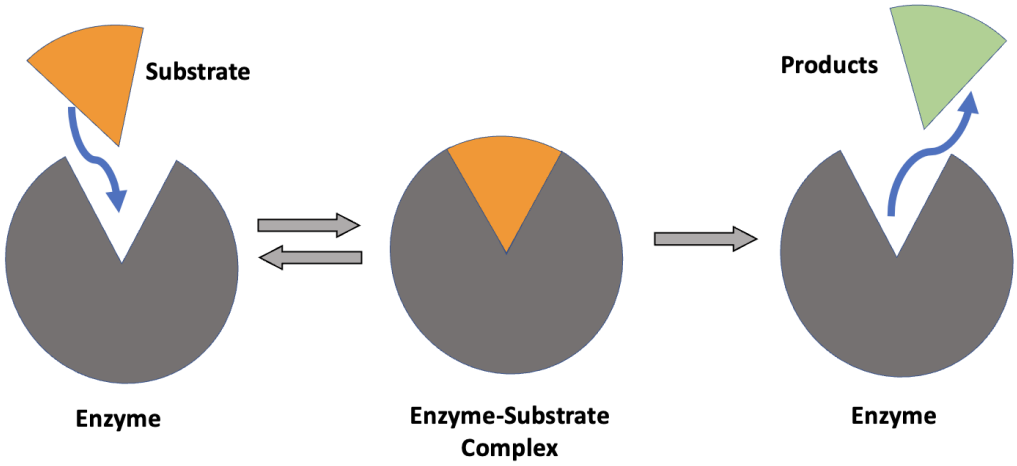


Figure 3.1: Michaelis-Menten Reaction System.

Writing the Michaelis-Menten reactions in their full forms gives the stoichiometry matrix

$$\begin{array}{cccc|c|cccc} S & E & ES & P & ! & S & E & ES & P & \\ \left[\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] & & & & & \left[\begin{array}{cccc} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right] & \begin{array}{l} S + E \overset{c_1}{!} ES \\ ES \overset{c_2}{!} S + E \\ ES \overset{c_3}{!} P + E \end{array} \end{array}.$$

The Michaelis-Menten stoichiometry change matrix (3.4) is given by

$$V^T = \begin{array}{cccc} S & E & ES & P \\ \left[\begin{array}{cccc} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right] & & & \end{array} \begin{array}{l} S + E \overset{c_1}{!} ES \\ ES \overset{c_2}{!} S + E \\ ES \overset{c_3}{!} P + E \end{array} \tag{3.6}$$

The Michaelis-Menten propensity functions are defined combinatorially as in Equation (3.5). For the duration of this work, unless otherwise stated, we take the reaction rates in computation to be $c_1 = 1, c_2 = 1, c_3 = 0.1$.

With four species and three reactions, the Michaelis-Menten system is relatively simple (but nonetheless relevant and realistic) and thus will form the basis of numerous examples throughout the remainder of this work.

3.2 Chemical Master Equation (CME) via Population Counts

Using these notions of reaction systems, we are interested in analyzing the state vector's evolution over time. Given an initial state vector $\mathbf{x}_0$, the chemical master equation (CME) seeks to compute the state of the system after a length of time t given as $\mathbf{x}(t)$. Since reactions firing is probabilistic, we cannot know for sure what state will be present at time t. Rather, we seek to determine the probability that the system will be in each state in the state space.

3.2.1 Overview

To analyze the change in states over time, let the probability that the system is in state $\mathbf{x}$ at time t be given by $P(\mathbf{x}, t)$. Define a column vector

$$\mathbf{p}(t) := [p_1(t), p_2(t), \ldots, p_M(t)]^T$$

in which each entry $p_m := P(\mathbf{x}_m, t)$, the probability of being in state $\mathbf{x}_m$ at time t. The change in this probability distribution for a single state $\mathbf{x}$ over time is given by

$$\frac{\partial P(\mathbf{x}, t)}{\partial t} = \sum_{k=1}^{M} \alpha_k(\mathbf{x} \quad \boldsymbol{\nu}_k) P(\mathbf{x} \quad \boldsymbol{\nu}_k, t) \quad \alpha_k(\mathbf{x}) P(\mathbf{x}) \tag{3.7}$$

Using this formulation of the CME, we can generate a transition rate matrix $\mathbf{A}$, which we refer to as the *CME Operator*, where element $a_{ij}, i \neq j$ is the propensity of changing

to state i when the system is currently in state j; these are given by

$$A(i,j) = \begin{cases} \alpha_k(x_j), & x_i = x_j + v_k \\ \sum_{k=1}^{M} \alpha_k(x_j), & i = j \\ 0, & \text{else} \end{cases} . \tag{3.8}$$

Starting with an initial probability vector p_0, the probabilities of being in each state at each period of time are found by solving the system of differential equations [7]

$$\begin{cases} \frac{d}{dt}p(t) &= A \quad p(t) \\ p(0) &= p_0 \end{cases} \tag{3.9}$$

which has exact solution

$$p(t) = \exp(tA)p(0). \tag{3.10}$$

One question that arises when performing this modeling is *what are the possible states?* Evidently, a large number of states would result in computations of high time and storage complexity. First, we will analyze a small finite set of states that we know are all possible reachable states from a given starting state; eventually, we will discuss the implications of realistic and arbitrarily large state spaces on the CME.

3.2.2 Michaelis-Menten Application: Finite State Space

To clearly illustrate the CME's functionality, this section presents a small example using the Michaelis-Menten reaction system in Table 3.1 defined previously. Suppose the system starts at state $x_1 = [2, 1, 0, 0]^T$. Then, all states possibly reached by the system and the corresponding CME operator are given by

$$
\begin{aligned}
\boldsymbol{x}_1 &= [2,1,0,0]^T \\
\boldsymbol{x}_2 &= [1,0,1,0]^T \\
\boldsymbol{x}_3 &= [1,1,0,1]^T \\
\boldsymbol{x}_4 &= [0,0,1,1]^T \\
\boldsymbol{x}_5 &= [0,1,0,2]^T
\end{aligned}
\qquad
\boldsymbol{A} = \begin{bmatrix}
2 & 1 & 0 & 0 & 0 \\
2 & 1.1 & 0 & 0 & 0 \\
0 & 0.1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1.1 & 0 \\
0 & 0 & 0 & 0.1 & 0
\end{bmatrix} \tag{3.11}
$$

A transition diagram, as introduced in Section 2.6.2, is useful for showing transitions between states. In this transition diagram, the states are $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_5$, and transitions occur via reactions R_1, R_2, and R_3. This is illustrated in Figure 3.2; here, we assume we start in state $\boldsymbol{x}_1$. Note that with each of these reactions' different probabilities of occurring, this is a stochastic automata model.

Beyond thinking of this model as simply transitioning from one state vector $\boldsymbol{x}$ to another, we may think of the four components of the state vector as the combination of four stochastic automata—one for each type of molecule—and we may see this as a stochastic automata network with its state space a hyper-rectangle or lattice from the cross product of the components. Chapter 4 shows how this can be conveniently formalized with tensors.

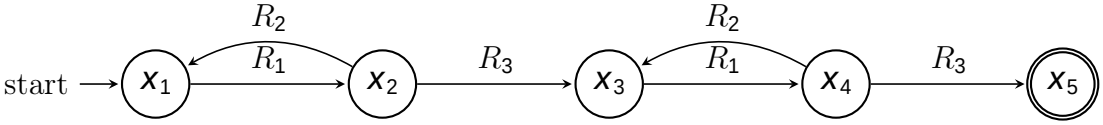


Figure 3.2: Michaelis-Menten 5-State Stochastic Automata Example

Using this information about state transitions, the CME can be employed to predict the probability of being in a particular state at time t. Assuming state $\boldsymbol{x}_1$ to be

the starting state gives initial probability vector

$$p(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The probability vectors after time t are then computed using Equation (3.10); for example, with $t = 10$ and $t = 100$, the resulting probability vectors are

$$p(10) = \begin{bmatrix} 0.1802 \\ 0.3485 \\ 0.2007 \\ 0.1742 \\ 0.0963 \end{bmatrix}, \qquad p(100) = \begin{bmatrix} 0.0005 \\ 0.0009 \\ 0.0128 \\ 0.0122 \\ 0.9736 \end{bmatrix}.$$

In this example, at time $t = 10$, the system is most likely to be in state x_2. However, by $t = 100$, the system will most likely be in state x_5. Figure 3.3 further illuminates the change in probabilities over time, plotting the probabilities (y-axis) of being in each state (x-axis) from times 0 to 100 in intervals of 20. As time progresses, the system appears to approach remaining in state x_5; this aligns with state x_5's status as a terminal state in Figure 3.2.

Using the states' probabilities at a given time, the system's *marginal probabilities*—probabilities for each individual species—at that time can be computed; these are the values of interest in most applied settings. Figure 3.4 shows the marginal probabilities for each species in the above example at each plotted time. Each sub-plot corresponds to a species $S, E, ES,$ or P; each row of subplots corresponds to a value of t. The x-axis holds the possible population counts for each species; the y-axis

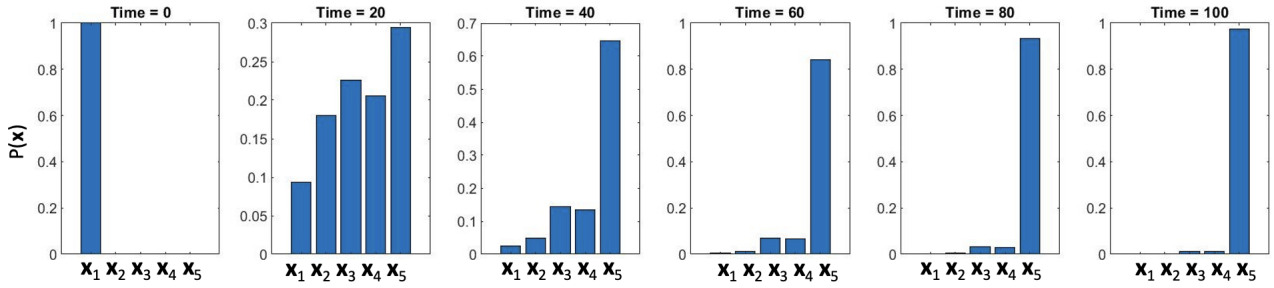


Figure 3.3: Probabilities of being in states $x_1 \dots x_5$ at times 0, 20, 40, 60, 80, and 100.

represents the corresponding probabilities. These marginal probability plots allow us to investigate how each individual species is likely to change over time. For instance, examining the first column of subplots shows how S shifts over time towards being most likely to have population count 0. At time $t = 100$, substrates are most likely to be at quantity 0, enzymes are most likely to be at quantity 1, enzyme-substrate complexes are most likely to be at quantity 0, and products are most likely to be at quantity 2. This provides a similar outcome to the state-wise analysis: these most-likely marginals correspond to the state $x = [0, 1, 0, 2]^T = x_5$, the most likely state at $t = 100$.

In this example, we assumed that we were dealing only with a finite and known list of reachable states. This assumption, however, is not realistic to what the real modeling problem looks like. It is known that without an imposed bound on the molecules' quantities, the state space can be arbitrarily large. A fundamental question of the CME arises in determining how to generate the CME operator in such cases.

3.2.3 State Space Truncation

The goal of the CME operator is to capture the transition from any one state to any other. However, without preexisting knowledge of a starting state, there are a theoretically countably infinite number of possible states that a system could be in. As such, we must truncate the state space, or put a bound on the quantity of each molecule we will represent in our model [7]. If the initial population count is known,

this can be done such that all possible states are captured, however, the state space can grow extremely large in realistic cases.

Each molecule $S_l, 1 \quad 1 \quad N$ is given a maximum population count $n_l \; 2 \; \mathsf{N}$ that may be present in any given state; i.e. $8x \; 2 \quad , 0 \quad x_l \quad n_l$. There are $I_l := n_l + 1$ possible values of each quantity x_l of S_l, so the state space consists of $j \;\; j = I_1 \;\; \ldots \;\; I_N$ possible states. The population count state space (3.1) is then given by the lattice or hyper-rectangle

$$= f(x_1, x_2, \ldots, \; x_N)^T \; 2 \; \mathsf{N}^N \, j \, 0 \quad x_l \quad n_l, \; l \; 2 \; f0, 1, \ldots, Ngg.$$

This results in $A \; 2 \; \mathsf{R}^{j \;\; j \;\; j \;\; j}$ and $p \; 2 \; \mathsf{R}^{j \;\; j}$ where $j \;\; j$ is extremely large in realistic applications. For example, if $I_l \quad 100$ and $J = 5$ species, then $j \;\; j$ is already of the order 10^{10}. For the following chapter the state space will refer to the truncated population count state space described here, unless stated otherwise.

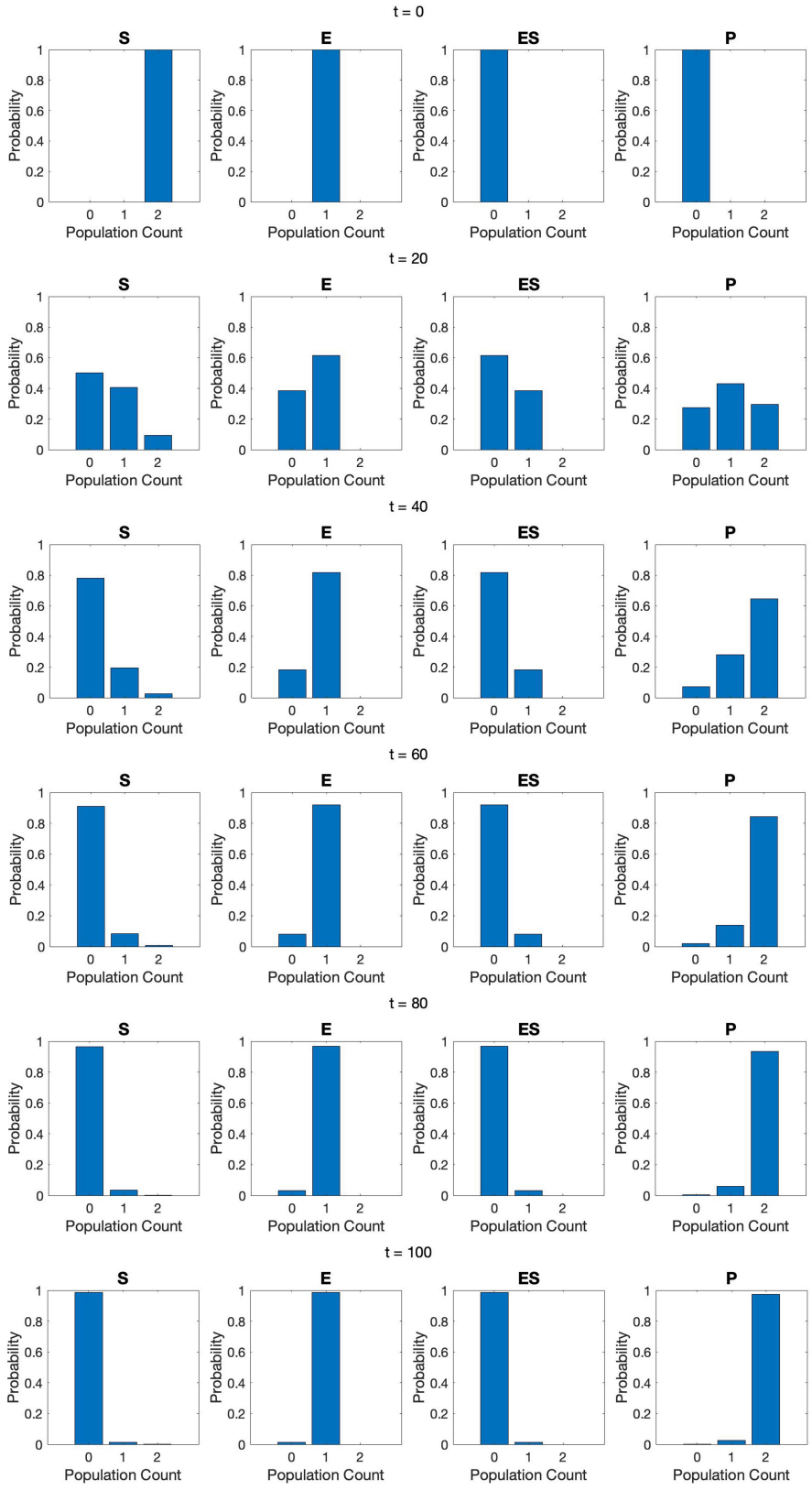


Figure 3.4: Marginal probabilities of each species' quantity at times t =0, 20, 40, 60, 80, 100. Each row corresponds to a time. Each column corresponds to a molecule. x-axis indicates population counts; y-axis gives corresponding probabilities.

CHAPTER 4

CHEMICAL MASTER EQUATION VIA TENSORS

As was just demonstrated in Chapter 3, with a large number of states and number of molecules (i.e. $j \;\; j, N \quad 1$), the problem of modeling biochemical reaction systems is high dimensional. This presents an opportunity for incorporating higher-order tensors [10, 7]. The bulk of this chapter is what was published in the faculty-reviewed paper [2].

Instead of enumerating the states and writing A as a matrix, one may instead write the CME operator as an order $2N$ tensor

$$\mathsf{A} \; 2 \; \mathsf{R}^{I_1 \quad I_N \;\; I_1 \quad I_N}$$

where the entry $\mathsf{A}_{x_1, \ldots, x_N, \, x_{N+1}, \ldots, x_{2N}}$ represents the propensity of transitioning from state $(x_{N+1}, \ldots, x_{2N})$ to state $(x_1, \ldots, x_N)$, such that each molecule S_l changes from quantity x_{N+l} to x_l.

The succeeding sections demonstrate the formulation of the matricized version of A, written as A, indexed by rows $(x_1, \ldots, x_N)$ and columns $(x_{N+1}, \ldots, x_{2N})$. These sections assume the combinatorial form of the propensity function α_k for each reaction R_k.

4.1 Classic Approach

In 2011, Hegland et al. [10] introduced a method for generating the CME operator as a sum of rank-one tensors; this work formed an important basis for similar tensor approaches. Notably, in 2014, Kazeev et. al [7] presented the incorporation of *tensor train decompositions* to the computational procedure of [10] while Dolgov and

Khoromskij [11] also used tensor train decompositions in application to the CME. These developments had the core benefit of improving the efficiency of storage and computation of the CME operator, which, as we have seen, can grow to be very large quickly enough that perpetually storing it in its entirety is neither appealing nor feasible. These works have sparked a substantial exploration of tensor train decompositions in the CME, such as the adaptive procedure in [12].

4.1.1 Algorithm

This section gives an overview of the algorithm to generate the matricized version of the CME operator using a sum of tensors as initiated by Hegland et al. [10]. We use the algorithmic version in Kazeev et al. [7], adapting notations to match those given throughout this work, where the resultant CME operator matrix A is generated by the sum

$$A = \sum_{k=1}^{M} (S_k \quad II) M_k, \tag{4.1}$$

The matrices II, S_k, M_k are of size $j \quad j \quad j \quad j$; II is the identity matrix; and S_k and M_k are generated by a series of Kronecker products as detailed below.

A reaction's M_k is a diagonal matrix storing the values of its propensity function α_k. That is, M_k at index $(\dot{u}(x), \dot{u}(x))$ is $\alpha_k(x)$ with all other values equal to zero. M_k may be computed by finding the vector $!^k$ such that $M_k = \mathrm{diag}(\omega^k)$. For each reaction R_k and species S_l, define the intermediate vector composed of binomial coefficients

$$!_l^k = \left(\alpha_k^{(l)}(0) \ldots \alpha_k^{(l)}(n_l)\right) = \left(\binom{0}{s_{kl}}, \quad , \binom{n_l}{s_{kl}}\right). \tag{4.2}$$

Using these, $!^k$ and M_k are given by

$$!^k = c_k \bigotimes_{l=1}^{N} !_l^k, \qquad M_k = \mathrm{diag}(!^k)$$

We are able to compute $\boldsymbol{M_k}$ as a series of Kronecker products due to the separability of the propensity function.

Similarly, a reaction's matrix $\boldsymbol{S}_k$ is defined by first creating matrices $\boldsymbol{S}_l^k$, $l = 1, \ldots, N$ (one for each species); these $\boldsymbol{S}_l^k$ matrices are shifted identity matrices in which the amount and direction shifted is dependent on the corresponding values of n_l and the stoichiometry change value v_{lk}. Specifically, for $v_{lk} = 0$, the corresponding $\boldsymbol{S}_l^k$ is the identity matrix. Else, $\boldsymbol{S}_l^k$ is given by

$$\boldsymbol{S}_l^k = \begin{cases} \begin{pmatrix} 0 & \cdots & 1 & & \\ & \ddots & & \ddots & \\ & & \ddots & & 1 \\ & & & \ddots & \vdots \\ & & & & 0 \end{pmatrix}, \text{ shifted up } |v_{lk}| \text{ rows if } v_{lk} < 0 \\ \begin{pmatrix} 0 & & & & \\ \vdots & \ddots & & & \\ 1 & & \ddots & & \\ & \ddots & & \ddots & \\ & & 1 & \cdots & 0 \end{pmatrix}, \text{ shifted down } v_{lk} \text{ rows if } v_{lk} > 0 \end{cases} \tag{4.3}$$

The formulation of the matrix is

$$\boldsymbol{S}_k = \bigotimes_{l=1}^{N} \boldsymbol{S}_l^k.$$

In [7], the quantized tensor train approximations of the $\boldsymbol{!}_k$ are formed before the Kronecker products are computed; although this was a crucial aspect of Kazeev et al. [7], this highly beneficial tensor compression step is noted as a comment in this work's presentation as our own focus rests in the formation of the operator via

Kronecker products. Further meanings behind these matrices and how they are computed are demonstrated through the pseudocode in Algorithms 4.1.1, 4.1.2, and 4.1.3. Refer to [7] for a more detailed explanation of this presentation of the approach; refer to [10] for a more detailed explanation of the original presentation of the approach.

Algorithm 4.1.1 Classic Approach [7, Alg. 1]

Data: *stoichmatrix; I; c*
// *stoichmatrix*; the stoichiometry matrix from Equation (3.3)
// *I* := $(I_1, \ldots, I_N)$; vector of mode sizes with *j* *j* = $\prod I$
// *c* := $(c_1, \ldots, c_I)$; vector of reaction rate constants
Result: *A*

1 *V* N M stoichiometry change matrix // create from *stoichmatrix*; refer to Equation 3.4
2 *II* identity matrix of size *j* *j*
3 *A* zero matrix of size *j* *j*
4 for k *1:*M do
5 S_k 1
6 w_k c_k
7 for l *1:*N do
8 S_l^k shifted identity matrix resulting from Algorithm 4.1.2
9 $!_l^k$ vector resulting from Algorithm 4.1.3
// Do kronecker products to generate S_k and M_k
10 S_k S_k S_l^k // matrix
11 $!^k$ $!^k$ $!_l^k$ // vector
12 end
13 M_k diag($!^k$)
14 *A* *A* + (S_k *II*) M_k
15 end

Algorithm 4.1.2 Create S_l^k

Data: I_l, v_{lk}
// Input data values come from Algorithm 4.1.1
Result: S_l^k
t $j v_{lk} j$
S_l^k zero matrix of size I_l
Replace $S_l^k(1 : I_l \quad t,\ 1 + t : I_l)$ with identity matrix of size t
if $v_{lk} > 0$ then
  S_l^k transpose of S_l^k
end

Algorithm 4.1.3 Create $! _l^k$

Data: I_l, s_{kl}
// Input data values come from Algorithm 4.1.1
Result: $! _l^k$
n_l I_l 1
$! _l^k$ $\left(\binom{0}{s_{kl}}, \ldots, \binom{n_l}{s_{kl}}\right)$ // propensity function values; see Equation (4.2)
// Kazeev et al. [7] achieve their compression by applying a QTT_Approx to $! _l^k$

4.1.2 Application Example: Michaelis-Menten

In this section, we expand on the small example given in Section 3.2.2; there, S and P both have maximum values of 2 while E and ES have maximum values of 1. Thus, we have $n_1 = n_4 = 2$ and $n_2 = n_3 = 1$. The size of the state space is given by $j\ j = 3 \quad 2 \quad 2 \quad 3 = 36$; the resulting matricized CME operator A will have a size of $36 \quad 36$. When using Equation (4.1), the generated matrix is shown in Figure 4.1; a blank indicates that the corresponding location has a zero value, while non-zero values are given in their positions. Although the state space has drastically increased and the index ordering has changed, careful examination of the matrix in (3.11) demonstrates that entries of the latter can be found in the former, though not necessarily with the order; this is expected, as that small state space is a subset of this one.

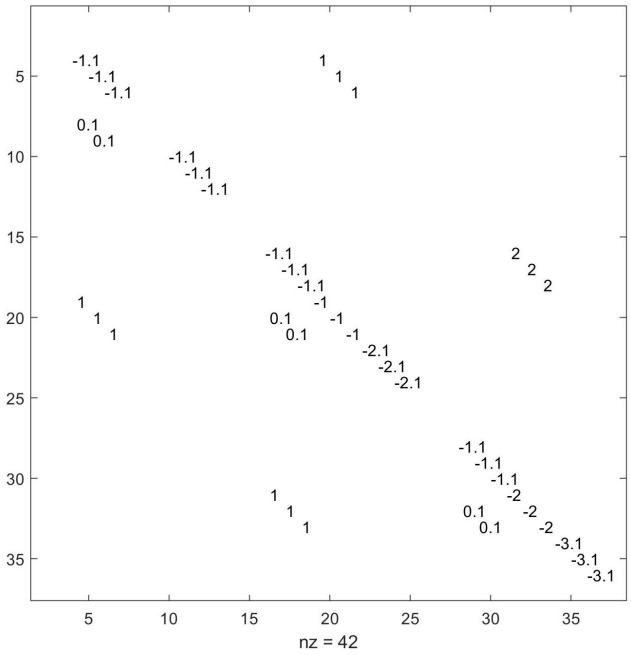


Figure 4.1: The shape and 42 non-zero values of the CME operator generated in Section 4.1.2. Columns sum to zero, except on boundary states where probability leaks occur.

4.2 SANs Approach

Prior to and separate from the approach summarized in Section 4.1.1 above, Wolf presented a method for computing the CME operator in 2007 [9]. Here, we further examine Wolf's methodology and show later that it is essentially equivalent to the more publicized classic tensor approach of [10] and made more efficient by [7], which is a striking discovery that has not been noted until now and was the main contribution of my work in [2].

4.2.1 Notations and Modeling Method

The formulation in [9] relies on the distinction between three types of molecules involved in the system of reactions: reactants, products, and catalysts.

- *Reactants*: Have a non-zero coefficient on the left side of the reaction (3.2); the set of reactant species for R_k are notated by $\mathbf{REA}(k)$.

- *Products*: Have a non-zero coefficient on the right side of the reaction (3.2); the set of product species for R_k are notated by $\mathbf{PRO}(k)$.

- *Catalysts*: Have non-zero coefficients on both the right and left-hand sides of the reaction (3.2) and a zero value in the change matrix (3.4); the set of catalyst species for R_k are given by $\mathbf{CAT}(k) = \mathbf{REA}(k) \setminus \mathbf{PRO}(k)$. [1]

Using these distinctions, Wolf generates the operator Q, that functions analagously to the transpose of the CME operator A from Equations (3.9) and (3.10); for more details, refer to [9]. The work in [9] was to show how to generate Q using the Kronecker product. The fundamental components of Wolf's method are presented here with slight modifications to some notations (primarily indexing); most notations are kept the same for ease of referring to the original work.

For each reactant $S_l \in \mathbf{REA}(k)$, Wolf defines the vector

$$\mathbf{dep}_l^k = \left(\binom{0}{s_{kl}}, \ldots, \binom{n_l}{s_{kl}} \right) \in \mathbb{N}^{(I_l)}. \tag{4.4}$$

For species S_l that are not reactants of R_k, Wolf instead defines

$$\mathbf{ind}_l^k = (1, \ldots, 1) \in \mathbb{N}^{I_l}. \tag{4.5}$$

Using these vectors, [9] defines $I_l \quad I_l$ matrices $\mathbf{Dep}_l^k(d)$ and $\mathbf{Ind}_l^k(d)$—these shift the corresponding **dep** and **ind** vectors to a d^{th} diagonal. For i, j such that $(i + d) = j$ where $j \quad i \; j \quad I_l$, and $\quad n_l \quad d \quad n_l$, the element at position (i, j) of $\mathbf{Dep}_l^k(d)$ is equal to the i^{th} element of $\mathbf{dep}_l^k$ and the element at position (i, j) of $\mathbf{Ind}_l^k(d)$ equals 1. All other entries are set to zero.

[1] The work in [9] makes the assumption without loss of generality that catalysts have equivalent stoichiometric coefficients on both sides of the reaction; we follow this assumption.

Using these definitions, [9] defines a matrix $E_l^{(k)}$ for each reaction R_k and species S_l given by

$$E_l^{(k)} = \begin{cases} \mathbf{Ind}_l^k(0) & \text{if } S_l \notin \mathbf{REA}(k) \cup \mathbf{PRO}(k) \\ \mathbf{Dep}_l^k(0) & \text{if } S_l \in \mathbf{CAT}(k) \\ \mathbf{Dep}_l^k(-s_{kl}) & \text{if } S_l \in \mathbf{REA}(k) \cap \mathbf{CAT}(k) \\ \mathbf{Ind}_l^k(s_{k,l+N}) & \text{if } S_l \in \mathbf{PRO}(k) \cap \mathbf{CAT}(k) \end{cases} \tag{4.6}$$

and a corresponding diagonal matrix $D_l^{(k)}$ given by

$$D_l^{(k)} = \mathrm{diag}(E_l^{(k)}\mathbb{1}), \qquad \mathbb{1} = (1,\ldots,1)^T \in \mathbb{N}^{I_l} \tag{4.7}$$

Wolf finally generates Q as

$$Q = \sum_{k=1}^{M} c_k \left(\bigotimes_{l=1}^{N} E_l^{(k)} \quad \bigotimes_{l=1}^{N} D_l^{(k)} \right). \tag{4.8}$$

This usage of sums of Kronecker products presents the opportunity for the potential addition of tensors train decompositions in a fashion analagous to that presented in [7]; we will comment more on this in the following sections.

4.2.2 Algebraic Version and Algorithm

Evidently, the method presented in [9] is largely based on if/else statements, particularly in the creation of the matrix $E_l^{(k)}$ in Equation (4.6). This work presents a the remodeling of these conditional statements into a more compact algebraic representation that is not only more readable but also useful for implementation of an automated algorithm in MATLAB to generate Q.

First, we examine the values for the diagonal shift d in $\mathbf{Dep}_l^k(d)$ and $\mathbf{Ind}_l^k(d)$ used in Equation (4.6) in the four different cases.

- If $S_l \notin \mathbf{REA}(k) \cup \mathbf{PRO}(k)$, then $s_{kl} = s_{k,N+l} = 0$. Thus, $s_{k,N+l} - s_{kl} = 0$.

- If $S_l \in \mathbf{CAT}(k)$, then $s_{kl} = s_{k,N+l}$. Thus, $s_{k,N+l} - s_{kl} = 0$.
- If $S_l \in \mathbf{REA}(k) \cap \mathbf{CAT}(k)$, then $s_{k,N+l} = 0$. Thus, $s_{k,N+l} - s_{kl} = -s_{kl}$.
- If $S_l \in \mathbf{PRO}(k) \cap \mathbf{CAT}(k)$, then $s_{kl} = 0$. Thus, $s_{k,N+l} - s_{kl} = s_{k,N+l}$.

Evidently, for all cases, the shift value is then given by $s_{k,N+l} - s_{kl}$. Note that if $S_l \notin \mathbf{REA}(k) \cup \mathbf{PRO}(k)$ or $S_l \in \mathbf{PRO}(k) \cap \mathbf{CAT}(k)$ then $s_{kl} = 0$; for any $n \in \mathbb{Z}$, $n \geq 0$, $\binom{n}{0} = 1$. For such S_l, the entries of $\mathbf{dep}_l^k$ would be all ones, which is the definition of $\mathbf{ind}_l^k$. Thus, the formulation of Equation (4.6) can be re-written using only **Dep** matrices and corresponding **dep** vectors as

$$E_l^{(k)} = \mathbf{Dep}_l^k (s_{k,N+l} - s_{kl}). \tag{4.9}$$

Algorithms 4.2.1, 4.2.2, and 4.2.3 are pseudo-codes, based on this re-formulation of the work in [9].

Algorithm 4.2.2 Create $\mathbf{dep}_l^k$

Data: I_l, s_{kl}

```
// Input data values come from Algorithm 4.2.1
```

Result: $\mathbf{dep}_k^l$

37 $n_l \leftarrow I_l - 1$

38 $\mathbf{dep}_k^l \leftarrow \left(\binom{0}{s_{kl}}, \ldots, \binom{n_l}{s_{kl}}\right)$ // binomial coefficients; refer to Equation (4.4)

Algorithm 4.2.1 SAN Approach

Data: *stoichmatrix; I; c*

// *stoichmatrix*; the M $2N$ stoichiometry matrix from Equation (3.3)

// *I* := (I_1, ..., I_N); vector of mode sizes with *j* *j* = $\prod$ *I*

// *c* := (c_1, ..., c_M); vector of reaction rate constants

Result: *A*

V N M stoichiometry change matrix // create from *stoichmatrix*; refer to Equation (3.4)

Q zero matrix of size *j* *j*

for k *1:* M do

  for l *1:* N do

    create $\mathbf{dep}_l^k$ using Algorithm 4.2.2 create $\boldsymbol{E}_l^{(k)}$ using Algorithm 4.2.3 create $\boldsymbol{D}_l^{(k)}$ using Equation (4.7)

    // Do kronecker products to get $\mathsf{E}^{(k)} = \bigotimes_{l=1}^{N} \boldsymbol{E}_l^{(k)}$ and $\mathsf{D}^{(k)} = \bigotimes_{l=1}^{N} \boldsymbol{D}_l^{(k)}$

    if j = 1 then

      $\mathsf{E}^{(k)}$ $\boldsymbol{E}_l^{(k)}$; $\mathsf{D}^{(k)}$ $\boldsymbol{D}_l^{(k)}$

    else

      $\mathsf{E}^{(k)}$ $\mathsf{E}^{(k)}$ $\boldsymbol{E}_l^{(k)}$; $\mathsf{D}^{(k)}$ $\mathsf{D}^{(k)}$ $\boldsymbol{D}_l^{(k)}$

    end

  end

  Q *Q* + c_i($\mathsf{E}^{(k)}$ $\mathsf{D}^{(k)}$)

end

Algorithm 4.2.3 Create $\boldsymbol{E}_l^{(k)}$

Data: $\mathbf{dep}_l^k$, v_{lk}

// Input data values come from Algorithm 4.2.1

Result: $\boldsymbol{E}_l^{(k)}$

I_l length of $\mathbf{dep}_j^i$ // is equal to I_l

t *j*v_{lk}*j*

$\boldsymbol{E}_l^{(k)}$ zero matrix of size I_l

if $v_{lk} > 0$ then

  Replace $\boldsymbol{E}_l^{(k)}$(1 : I_l t , 1 + t : I_l) with diag($\mathbf{dep}_l^k$(1 : I_l t))

else

  Replace $\boldsymbol{E}_l^{(k)}$(1 + t : I_l , 1 : I_l t) with diag($\mathbf{dep}_l^k$(1 + t : I_l))

end

4.2.3 Application Example: Michaelis-Menten

We use the method from Sections 4.2.1 and 4.2.1 to compute $\boldsymbol{Q}$ for the same state space given in Section 4.1.2, where S and P have the three possible values from 0 to 2 and E and ES have the two possible values 0 and 1; the resulting matrix $\boldsymbol{Q}$ is given in Figure 4.2.

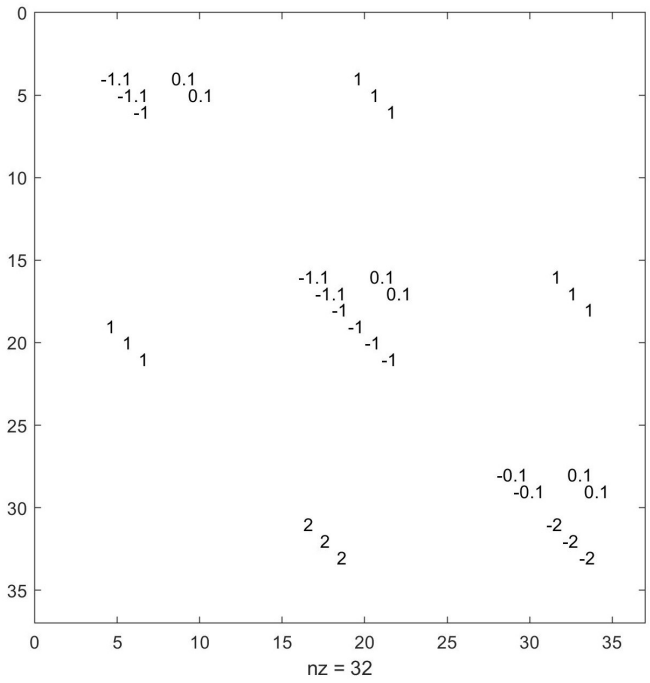


Figure 4.2: The shape and 32 non-zero values of the CME operator from Section 4.2.3. Rows sum to zero.

It is immediately clear that the matrices in Figures 4.1 and 4.2 are not the same—they have different numbers of non-zero elements. Upon closer inspection, however, it is apparent that off-diagonal elements of $\boldsymbol{Q}^T$ are equal to the off-diagonal elements of $\boldsymbol{A}$. In the following section, we further explore these differences and similarities; it turns out that this off-diagonal similarity holds true in the general case.

4.3 Methodology Comparison

In the above section, we saw that the two generation methods led to different, but similar, CME operators. A key difference between them is that the sum of each

column of the transpose of $\boldsymbol{Q}$ in Section 4.2.3 is zero, whereas this is not the case for $\boldsymbol{A}$ in Section 4.1.2. This indicates that, when using $\boldsymbol{Q}$, each resulting probability vector will sum to 1; on the other hand, using $\boldsymbol{A}$ in Section 4.1.1, the resulting probability vectors may sum to a value less than 1—that is, we see 'probability leaks'.

Probability Leaks

The source of the probability leaks stems from truncating the state space. To clearly see why this is the case, suppose that a Michaelis-Menten system had starting state $[1, 0, 2, 2]$ —it would then be possible to transition to state $[2, 1, 1, 2]$ via R_2. Say, however, that the model uses the state space of Sections 4.1.2 and 4.2.3, which limits the quantity of ES to 1. This possible state $[2, 1, 1, 2]$ is not found in the state space and thus the propensity to transition to it from the starting state $[1, 0, 2, 2]$ is not included in the CME operator. As a result, the computed probability vector should, intuitively, not sum to one, as not all possible states are represented.

However, Equation (4.3), through the creation and subtraction of the diagonal $\boldsymbol{D}_l^{(k)}$ matrices, forces the rows to sum to zero and thus eliminates the probability leaks. When examining probability leaks, we refer to possible states outside of the state space as *sinks*. For a given initial state, as the size of the state space increases the classic tensor approach, the amount of probability lost to the sinks will decrease.

Equivalence of Non-Diagonal Elements

As was shown in the Michaelis-Menten computational examples, there appeared to be numerical equivalency between the off-diagonal elements of each method's CME operator. This work demonstrates a proof that the off-diagonal elements resulting from the classic method [10, 7] are identical to the transpose of those resulting from the SAN method [9].

Recall that each algorithm is performed iteratively—for each reaction, several matrices are generated and summed together. Additionally, recall that the classic methodology's algorithm is summarized by the equation

$$A = \sum_{k=1}^{M} (S_k \quad II) M_k$$

where A is the CME operator, with

$$S_k = S_1^k \qquad S_N^k, \qquad M_k = \text{diag}(c_k \quad !_1^k \qquad !_N^k),$$

as described in Section 4.1.1. The SAN methodology of [9] is summarized by

$$Q = \sum_{k=1}^{M} c_k \left(\bigotimes_{l=1}^{N} E_l^{(k)} \quad \bigotimes_{l=1}^{N} D_l^{(k)} \right).$$

where Q functions analagously to the CME operator in transposed form, as is described in Section 4.2. Since M_k and $\bigotimes_{l=1}^{N} D_l^{(k)}$ are diagonal matrices, demonstrating the equality of the off-diagonal elements of A and Q^T can be reduced to demonstrating that

$$\sum_{k=1}^{M} S_k M_k = \left(\sum_{k=1}^{M} c_k \bigotimes_{l=1}^{N} E_l^{(k)} \right)^T = \sum_{k=1}^{M} \left(c_k \bigotimes_{l=1}^{N} E_l^{(k)} \right)^T. \tag{4.10}$$

If for each $k \in [1, M]$ we have

$$S_k M_k = \left(c_k \bigotimes_{l=1}^{N} E_l^{(k)} \right)^T,$$

then Equation (4.10) will hold true. Thus, in expanded form, we aim to show that

$$(S_1^k \qquad S_N^k) \quad \text{diag}(c_i \quad !_1^k \qquad !_N^k) = (c_i \quad E_1^{(k)} \qquad E_N^{(k)})^T.$$

Owing to properties of the Kronecker product, this can be re-written as

$$(S_1^k \quad \text{diag}(\mathbf{!}_1^k) \qquad (S_N^k \quad \text{diag}(\mathbf{!}_N^k)) = (E_1^{(k)})^T \qquad (E_N^{(k)})^T.$$

Thus, the proof is complete if we show that for any reaction R_k and species S_l

$$S_l^k \quad \text{diag}(\mathbf{!}_l^k) = (E_l^{(k)})^T. \tag{4.11}$$

To prove Equation (4.11), note that, as was mentioned previously, the vector $\mathbf{dep}_l^k$ for reaction R_k and molecules S_l used in creating $E_l^{(k)}$ is equivalent to $\mathbf{!}_l^k$; we have

$$\mathbf{dep}_l^k = \mathbf{!}_l^k = \left(\begin{pmatrix} 0 \\ s_{kl} \end{pmatrix}, \quad , \begin{pmatrix} n_l \\ s_{kl} \end{pmatrix} \right). \tag{4.12}$$

From this point, we can look at the problem under two different lights: expanding matrices for comparison (more intuitive) vs. solving analytically (more compact). These are two approaches to the same proof; we discuss each in turn.

Matrix Expansion: Note that $E_l^{(k)}$ 2 $\mathsf{R}^{I_l \quad I_l}$ is given by

$$
\begin{cases}
I_l \quad t \quad ! \begin{bmatrix} 0 & (\mathbf{dep}_l^k)_1 & & \\ & & \ddots & \\ & & & (\mathbf{dep}_l^k)_{I_l \quad t} \end{bmatrix}, \; S_l \in \mathbf{PRO}(k) & \text{(4.13a)} \\
1 + t \quad ! \begin{bmatrix} 0 & & & \\ \vdots & & & \\ (\mathbf{dep}_l^k)_{1+t} & & & \\ & \ddots & & \\ & & (\mathbf{dep}_l^k)_{I_l} & 0 \end{bmatrix}, \; S_l \in \mathbf{PRO}(k) & \text{(4.13b)}
\end{cases}
$$

where, consistent with the notation in Algorithms 4.1.2 and 4.2.3, t refers to $|v_{lk}|$.

Now, note that $S_l^k \in \mathbb{R}^{I_l \quad I_l}$ is given by

$$\begin{cases} 1 + t \;\; ! \begin{bmatrix} 0 & & & \\ \vdots & & & \\ 1 & & & \\ & \ddots & & \\ & & 1 & 0 \end{bmatrix}, & I_l \;\; t \;\; \# \;\; S_l \; 2 \; \mathbf{PRO}(k) \quad (4.14a) \\ I_l \;\; t \;\; ! \begin{bmatrix} 0 & 1 & & \\ & & \ddots & \\ & & & 1 \\ & & & \vdots \\ & & & 0 \end{bmatrix}, & 1 + t \;\; \# \;\; S_l \; 2 \; \mathbf{PRO}(k) \quad (4.14b) \end{cases}$$

Multiplying S_l^k from (4.14) by $\mathrm{diag}(!_l^k) = \mathrm{diag}(\mathbf{dep}_l^k)$ from Equation (4.12) yields the matrix in $\mathsf{R}^{I_l \;\; I_l}$ given by

$$
\begin{cases}
\begin{array}{c} I_l \quad t \\ \# \\ 1+t \end{array} \; ! \begin{bmatrix} 0 \\ \vdots \\ (!_l^k)_1 & & \\ & \ddots & \\ & & (!_l^k)_{I_l \; t} \end{bmatrix}, \quad S_l \in \mathbf{PRO}(k) & (4.15a) \\
\begin{array}{c} 1+t \\ \# \\ I_l \quad t \end{array} \; ! \begin{bmatrix} 0 & (!_l^k)_{1+t} & & \\ & & \ddots & \\ & & & (!_l^k)_{I_l} \\ & & & \vdots \\ & & & 0 \end{bmatrix}, \quad S_l \in \mathbf{PRO}(k) & (4.15b)
\end{cases}
$$

We see from Equations (4.13) and (4.15) that

$$S_l^k \, \mathrm{diag}(!_l^k) = (E_l^{(k)})^T.$$

Thus, we see that the off-diagonal elements of A are equivalent to the off-diagonal elements of Q^T.

Solving Analytically: From Equations (4.9) and (4.12), proving the equality in Equation (4.11) is equivalent to demonstrating

$$
\begin{aligned}
S_l^k \quad \mathrm{diag}(!_l^k) &= (\mathbf{Dep}_l^k(s_{k,N+l} \quad s_{kl}))^T \\
&= (\mathrm{diag}(\mathbf{dep}_l^k)\,\mathbf{Ind}_l^k(s_{k,N+l} \quad s_{kl}))^T \\
&= (\mathbf{Ind}_l^k(s_{k,N+l} \quad s_{kl}))^T(\mathrm{diag}(\mathbf{dep}_l^k))^T
\end{aligned}
$$

$$= (\mathbf{Ind}_l^k\ (s_{k,N+l} \quad s_{kl}))^T \mathrm{diag}(\mathbf{dep}_l^k)$$
$$= (\mathbf{Ind}_l^k\ (s_{k,N+l} \quad s_{kl}))^T \mathrm{diag}(\ !\ _l^k) \tag{4.16}$$

Evidently, this simplifies to demonstrating that

$$S_l^k = (\mathbf{Ind}_l^k\ (s_{k,N+l} \quad s_{kl}))^T. \tag{4.17}$$

By definition $v_{k_l} = s_{k,N+l} \quad s_{kl}$. Thus, from their respective definitions, S_l^k and $\mathbf{Ind}_l^k\ (s_{k,N+l} \quad s_{kl})$ are each shifts of the $I_l \quad I_l$ identity matrix by magnitude v_{l_k} but in opposite directions; Equation (4.17) follows as true. It follows that the off-diagonal elements of A are indeed equivalent to the off-diagonal elements of Q^T.

CHAPTER 5

CHEMICAL MASTER EQUATION VIA REACTION COUNTS

As has been seen thus far, storing and computing the population counts and their associated propensities can get extremely computationally complex very quickly. An alternative method is to instead count the number of each reaction that has occurred; under certain conditions, this technique has the potential to lessen the computational burden.

5.1 Set-Up

In dealing with reaction counts, several additions to the set-up of the chemical master equation seen before are necessary. First, and perhaps most evident, is the reformulation of the state space. Previously, we dealt primarily with , the population count state space. Instead, the state space must now represent the counts of each reaction that has occurred; call this state space $_R$. In a system with M reactions, $_R$ N^M. A state $\mathbf{r}$ 2 $_r$ is defined by

$$\mathbf{r} := (r_1, r_2, \ldots, r_M)^T \; 2 \; \mathsf{N}^M$$

where r_i represents the number of times reaction R_i has fired. To map from reaction counts back to population counts, define a mapping : $_R$! . Taking $\mathbf{x}_0$ 2 as the initial population count state,

$$(\mathbf{x}_0, \mathbf{r}) = \mathbf{x}_0 + V\,\mathbf{r} \tag{5.1}$$

where V is the stoichiometric change matrix initially defined in Section 3.1.

Example: Growth and Decay Process

Define a simple reaction system representing growth and decay processes for a molecule S by the two reactions shown in Table 5.1

$$R_1 : S \xrightarrow{\theta_1} ;$$
$$R_2 : ; \xrightarrow{\theta_2} S$$

Table 5.1: Growth Decay Reaction System.

The stoichiometric matrix is $V = \begin{bmatrix} -1 & 1 \end{bmatrix}$ and the reaction count state space $_R$ is in $\mathbb{N}^2$.

Suppose we start with 15 S molecules. Additionally, suppose there have been three growth reactions and two decay reactions leaving 16 of molecule S. This is represented by taking $\mathbf{x}_0 = [15]$ and $\mathbf{r} = [2, 3]^T$. Then

$$(\mathbf{x}^{(0)}, \mathbf{r}) = [15] + [-1, 1] \begin{bmatrix} 2 \\ 3 \end{bmatrix} = [16].$$

5.2 Solving the CME via Reaction Counts

As in the CME via population counts, the goal of the CME via reaction counts is to determine the probabilities of being in certain states after a time t. Denote the probability of a system being in state $\mathbf{r}$ at time t by $P(\mathbf{r}, t)$. Then the probability of being in state $\mathbf{r}$ at time t can be derived by the equation

$$\partial_t P(\mathbf{r}; t) = \sum_{k=1}^{M} \alpha_k(\mathbf{x}_0 + V(\mathbf{r} - \mathbf{I}_k))P(\mathbf{r} - \mathbf{I}_k; t) - \sum_{k=1}^{M} \alpha_k(\mathbf{x}_0 + V\mathbf{r})P(\mathbf{r}; t) \tag{5.2}$$

where $\mathbf{I_k}$ is the identity vector with a one at index k and α_k is the propensity function as defined in Section 3.1. Define a column vector

$$q(t) := [q_1(t), q_2(t), \ldots, q_{j\ _{Rj}}(t)]^T$$

in which each entry $q_m := P(\mathbf{r}_m, t)$, the probability of being in state $\mathbf{r}_m$ at time t.

Using this formulation of the reaction count based CME, we are interested in generating a transition rate matrix A^R, which we refer to as the *CME reaction count operator*, where element $a_{ij}^R, i \neq j$ is the propensity of changing to state i when the system is currently in state j and defined as

$$A^R(i,j) = \begin{cases} \alpha_k(x_0 + V r_j), & r_i = r_j + l_k \\ \sum_{k=1}^{M} \alpha_k(x_0 + V r_j), & i = j \\ 0, & \text{else} \end{cases} . \tag{5.3}$$

Notice that in each time step, we allow only one reaction to occur. As a result, only one entry of a state r may change. Specifically, this singular changing entry *must increase by one*—unlike population counts, reaction counts may never decrease as time progresses. This "push-forward" nature of the reaction count state space lends itself a nice property of the CME reaction count operator: when states are arranged properly, A^R is lower triangular, which has many potential computational benefits.

Using the operator A^R in solving the reaction count CME is very similar to the population count system. Starting with an initial probability vector q_0, the probabilities of being in each state at each period of time are found by solving the system of differential equations

$$\begin{cases} \frac{d}{dt} q(t) &= A^R \ q(t) \\ q(0) &= q_0 \end{cases} \tag{5.4}$$

which has exact solution

$$q(t) = \exp(tA^R) q(0). \tag{5.5}$$

Although the result of the reaction count CME gives probabilities of how many reactions have occurred by time t, we are interested in an eventually learning population count probabilities. To do so, can be applied to each state $\boldsymbol{r}$; probabilities for reaction count states that lead to equivalent population count states are summed together.

In moving forwards in the analysis of the reaction count CME, we focus on a truncated reaction count state space. Just as in the preceding chapters in which we examined the state spaces with each species S_l having upper bound population count n_l, the state spaces we focus on with regards to reaction counts allow each reaction R_k to occur at an upper bound of n_k^Rf times.

5.3 Tensor-Based Reaction Count CME Operator

As was discussed in Chapter 4, the formulation of an efficient tensor train compression of the chemical master equation arises from the ability to write the CME operator as a composition of Kronecker products. To the best of knowledge at the time of this work, no tensor or Kronecker product formulation of the reaction count CME operator has been discussed. Despite the similarities between the population count and reaction count CME, such a formulation is non-trivial. The population count Kronecker product formulation rests in the separability of the propensity functions $\alpha_k(\boldsymbol{x})$ for $\boldsymbol{x}$ 2 . However, in the reaction count state space, the propensity functions are still based in population counts; the propensity function for a state $\boldsymbol{r}$ 2 $_R$ given by $\alpha_k(\boldsymbol{x_0} + \boldsymbol{V}\, \boldsymbol{r})$ is not separable for $\boldsymbol{r}$. This work presents a novel formulation of the CME reaction count operator based in Kronecker products and Kronecker sums.

Similarly to in the formulation of the CME operator of [10, 7] as discussed in Section 4.1.1, the formulation of $\boldsymbol{A}_r$ can be summarized by

$$\boldsymbol{A}^{\boldsymbol{R}} = \sum_{k=1}^{M} (\boldsymbol{S}_k^{\boldsymbol{R}} \quad \boldsymbol{II})\, \boldsymbol{M}_k^{\boldsymbol{R}} \tag{5.6}$$

where S_k^R, M_k^R, I are matrices of size $j \; _R j \quad j \; _R j$ as detailed below and I is still the identity matrix.

Similarly to the presentation in Section 4.1.1, S_k^R is the Kronecker product of intermediate shift matrices $S_{k,\,k_2}^R \; 2 \; \mathbb{R}^{n_{k2}^R \;\; n_{k2}^R}$ for $1 \quad k_2 \quad M$. If $k = k_2$ then $S_{k,k2}^R$ is the identity matrix shifted down by one; else, $S_{k,k2}^R$ is the identity matrix. Computing the Kronecker product gives

$$S_k^R = \bigotimes_{k_2=1}^{R} S_{k,k2}^R \tag{5.7}$$

Matrix M_k^R is a diagonal matrix storing the propensity function values for the associated reaction rate states in lexicographical order. That is, entry of $M_k^R(\dot{u}(r), \dot{u}(r)) := \alpha_k(x_0 + V r)$; all other entries are zero. The matrix M_k^R may be computed by finding $!^{R^k}$ such that $M_k^R = \text{diag}(!^{R^k})$. To aid in computation, define for each species S_l the vector $f_l \; 2 \; \mathbb{R}^{j \;\; _R j}$ to be the vector storing the population count of S_l at each reaction state in lexicographical order. More specifically, for each $i \; 2 \; \{1 \ldots j \;\; _R j\}$, take the unique r such that $\dot{u}(r) = i$. Define the corresponding $x_i := x_0 + V^T r_i$. Then $f_l = (x_l^{(1)}, \ldots, x_l^{(j \;\; _R j)})$ This vector is computed via iterative Kronecker sums as shown in Algorithm 5.3.1.

Algorithm 5.3.1 Create f_l

```
f_l  0
for k  1:M do
    f_l  f_l  v_lk[0, ..., n_k^R]
end
```

For each reaction, let $!_l^{R^k}$ for $1 \quad l \quad N$ be the vector of length $j \; _R j$ given by applying $\alpha_k^{(l)}$ to each element of f_l, i.e. $!_l^{R^k} := \left(\alpha_k^{(l)}(f_{l1}) \ldots \alpha_k^{(l)}(f_{l_j \;\; _R j})\right)$. Finally, Letting $\;$ represent the element-wise product, $!^{R^k}$ and M_k^R are computed as

$$!^{R^k} = c_k \, !_l^{R^1} \quad \ldots \quad !_N^{R^1}, \qquad M_k^R = \text{diag}(!^{R^k})$$

Note that the necessity of transferring between reaction counts and population counts amid the computation of the CME reaction count operator substantially ads to its complexity.

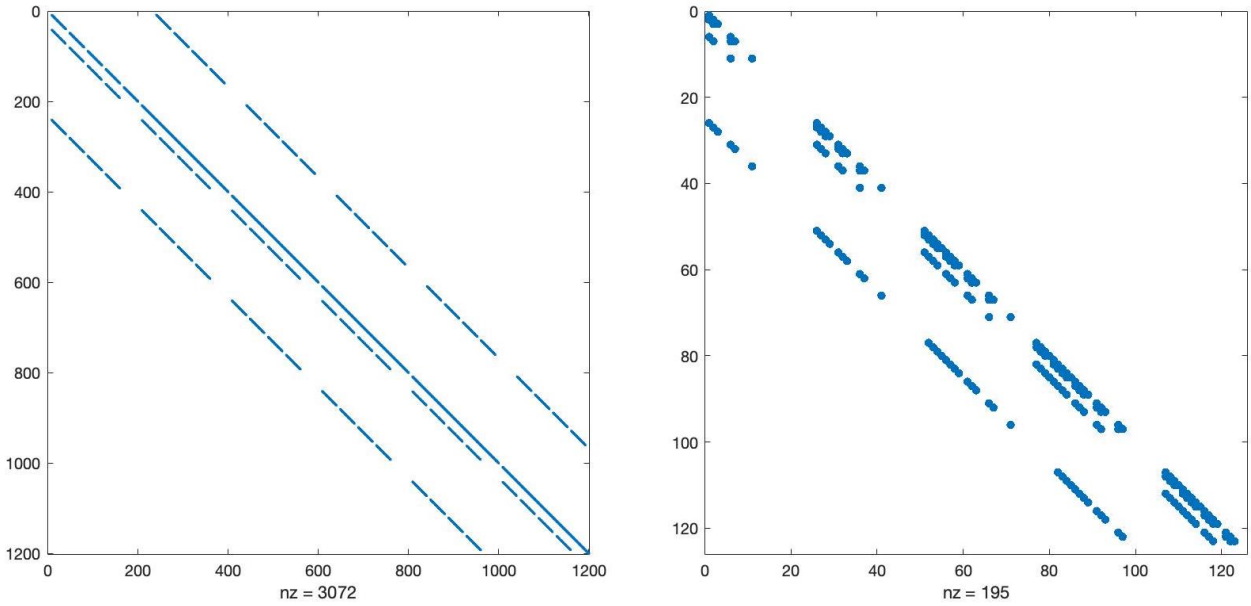
5.4 Computational Example

To analyze the output of the output of the CME via reaction counts compared to population counts, we introduce a computational example based on the Michaelis-Menten system introduced in Section 3.2.

Suppose that the initial population state is $x_0 = [3, 2, 2, 2]^T$. Let the upper bounds on population counts be $n_1 = 6, n_2 = 5, n_3 = 5, n_4 = 8$ for S, E, ES, P respectively. These upper bounds are set such that for the given x_0, all reachable states are included in the state space. Let the upper bounds on reaction counts be $n^R := n_1^R = n_2^R = n_3^R = 5$ for each R_1, R_2, R_3. Here, the reaction count state space is substantially smaller than the population count state space with $|_R| = 125$ and $| | = 1440$. Figure 5.1 shows the shapes of the associated population count CME operator A and reaction count CME operator A^R. Figure 5.1b illustrates the lower triangular shape of the affiliated CME reaction count operator.

Let p_0 and q_0 be the population count and reaction count probability vectors assigning 100% probability of beginning in state x_0. For a given time t with $p(t) = \exp(At)p_0$ and $q(t) = \exp(A^R t)q_0$, let $\sum p(t)$ and $\sum q(t)$ represent the sums of the resulting probability vectors.

Figure 5.2 illustrates the marginal population count probabilities for the case where $t = 0.5$ computed by means of the population count CME (top) and reaction count CME (bottom). Although the two plots are visually effectively identical, analysis of the resulting probability vectors reveals that the two methods produce slightly different results. Summing the probability vectors reveals that $\sum p(0.5) = 1.0$ whereas $\sum q(0.5) = 0.9859$, indicating that the reaction count method has led to some probability leaks. This indicates that the selected upper bounds on reaction firings



(a) A: Population count CME operator. (b) A^R: Reaction count CME operator

Figure 5.1: Example shapes of population count and reaction count CME operators modeling the same system.

were not large enough for the time window 0.5. It appears that by $t = 0.5$, there is a notable probability that at least one reaction has fired more than 5 times thus causing a transition to a state not modeled by our state space.

Table 5.2 gives more results demonstrating the probability leaks occurring in the reaction count probability vector using the same initial state $x_0 =$ for four different state spaces. The first column n^R gives the upper bound on reaction counts; the state space is size $j \quad _R j^3$. The following four columns give the sum of the resulting probability vector $q(t)$ given at $t = 0.5, t = 1, t = 2.5$, and $t = 5$. For the small state space with $n_R = n_1^R = n_2^R = n_3^R = 5$, there are already substantial probability leaks by $t = 1$. While setting $n_R = n_1^R = n_2^R = n_3^R = 20$ allows better modeling up to $t = 5$, the corresponding state space is extremely large.

While it is convenient in some capacities that reaction count states are always increasing, this "push-forward" tendency means that as time progresses past even a

n^R	$\|\mathbf{j}_R\|$	$\sum q(t)$			
		t = 0.5	t = 1	t = 2.5	t = 5
5	5^3	0.9859	0.7456	0.0341	0
10	10^3	1.0	0.9999	0.7673	0.0414
15	15^3	1.0	1.0	0.9973	0.4944
20	20^3	1.0	1.0	1.0	0.9377

Table 5.2: Computational results for CME probability leaks via reaction counts with $x_0 = [3, 2, 2, 2]^T$. A value of $\sum q(t)$ closer to 1 indicates less probability leak.

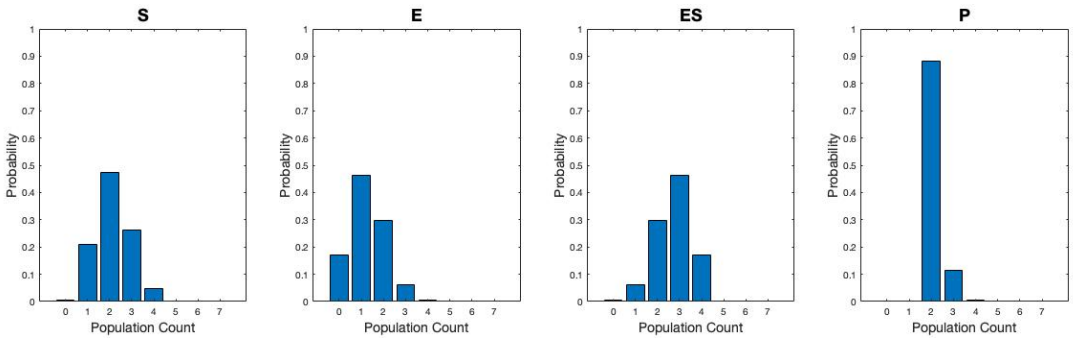
.

very small increment, accurate modeling requires an extremely large state space; this growth in size is likely to outweigh any potential benefits gained from the lower triangular nature of A^R.

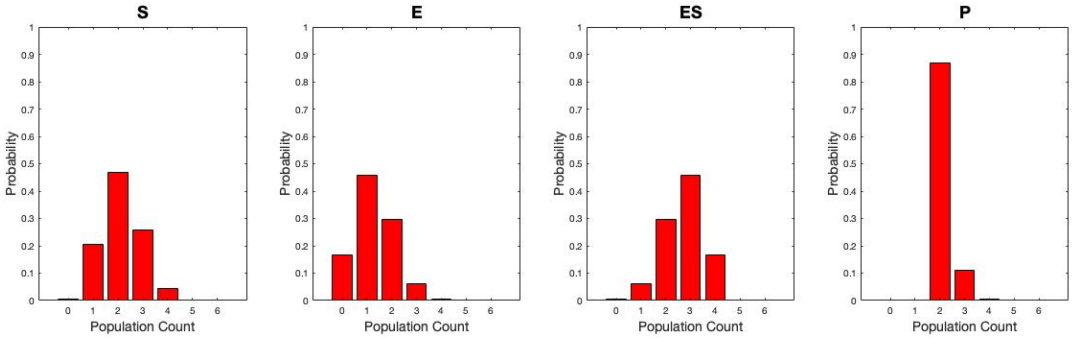
However, there is a potential modification that is promising in using the reaction count CME for any time interval. The following section explains how we may seek to take advantage of the fact that reaction counts never decrease—once a reaction count state has occurred, it no longer needs to be modeled.

5.5 Adaptive Solution to the Reaction Count CME

The introduction of an adaptive solution to the reaction count CME aims to take advantage of the fact that within a small time step, only a portion of the states in a larger state space are viable. This means that, within small time intervals, we may look at a comparatively small window of states. Take time sub-intervals $\tau_i, 1 \quad i \quad K$ such that $\sum_{i=1}^{K} \tau_i = t$. Then $e^{tA^R}q(0) = e^{(\tau_K + \ldots + \tau_1)A^R}q(0) = e^{\tau_K A^R} \quad e^{\tau_1 A^R}q(0)$. Letting $q_0 = q(0)$, define intermediate probability vectors $q_i := e^{\tau_i A^R}q_{i\ 1}$ that are easier to deal with than the original computation of $e^{tA^R}q(0)$. Furthermore, at each τ_i, we can determine a corresponding $A^R_{\tau_i}$ that contains transition propensities for the small subset of states relevant in the time interval τ_i. While this adaptivity will introduce intricacies in re-indexing and re-sizing the CME operators and probability vectors, there is likely to be significant payoff.



(a) Marginal probabilities computed via population count method.



(b) Marginal probabilities computed via the reaction count method.

Figure 5.2: Marginal population count probabilities computed via population and reaction counts for Michaelis-Menten at t = 0.5. Population count marginals in 5.2b are recovered from the distribution computed by the reaction count method.

Similar adaptions have been demonstrated for population counts [6, 5, 12] However, determining the relevant states for τ_i seems to be a more challenging in the case of population counts as states may be returned to once they have occurred. The possibility of an adaptive reaction count CME opens the doors for promising future investigations.

CHAPTER 6

CONCLUSIONS

The chemical master equation (CME), which models the interactions between species in a system of biochemical reactions, is a perfect avenue for the applications of higher order tensors and their efficient decompositions (particularly the tensor train and quantized tensor train decompositions).

This work reviewed two methods for using tensors to develop solutions to the chemical master equation; the first is the classic method presented by Hegland et al. in *On the Numerical Solution of the Chemical Master Equation with Sums of Rank One Tensors* and further explored by Kazeev et al. in *Direct Solution of the Chemical Master Equation Using Quantized Tensor Train*; the second is that presented by Wolf in her work *Modelling of Biochemical Reactions by Stochastic Automata Networks.* Though the classic formulation has received substantially more recognition in the literature, we proved key similarities between the approaches. The close relationship between stochastic automata networks and higher-order tensors motivates exploring higher-order tensors in other stochastic automata problems and continuous time Markov processes outside of chemical reaction systems.

Additionally, this work explored the potential for modeling the CME based on reaction counts as opposed to population counts. Although modeling the CME via a reaction count state space is, at the surface, a very similar to the population count problem, many challenging intricacies are introduced. Even with states based on reaction counts, population count information is needed at each time step to compute transition propensities; these computations must be implicitly incorporated. A successful incorporation of the QTT into the Kronecker product reformulation of the

reaction count CME has many potential benefits: the truncated reaction count state space itself is simpler and its CME operator has a lower triangular form because reaction counts do not decrease. Further investigating the potential for an adaptive QTT approach to the reaction count CME is an exciting avenue for future work that promises to be impactful in modeling biochemical reaction systems.

REFERENCES

[1] Tamara Kolda and Brett Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009.

[2] Marie Neubrander. Stochastic automata networks and tensors with application to chemical kinetics. *SIAM Undergraduate Research Online*, 13, 07 2020.

[3] William Stewart, Karem Atif, and Brigette Plateau. The numerical solution of stochastic automata networks. *European Journal of Operational Research*, 86:503–525, 1995.

[4] Ivan Oseledets. Tensor-train decomposition. *SIAM J. Scientific Computing*, 33:2295–2317, 01 2011.

[5] Huy Vo. *KRYLOV APPROXIMATION AND MODEL REDUCTION METHODS FOR SOLVING THE CHEMICAL MASTER EQUATION*. PhD dissertation, The University of Alabama, 2017.

[6] Trang Dinh and Roger B Sidje. An adaptive solution to the chemical master equation using quantized tensor trains with sliding windows. *Physical Biology*, 17, 10 2020.

[7] Vladimir Kazeev, Mustaga Khammash, Michael Nip, and Christoph Schwab. Direct solution of the chemical master equation using quantized tensor train. *PLOS Computational Biology*, 10(3), 2014.

[8] Alfred Aho, Monica Lam, Ravi Sethi, and Jeffrey Ullman. *Compilers: Principles, Techniques, and Tools*. Pearson Addison–Wesley, 2nd edition, 2007.

[9] Verena Wolf. Modelling of biochemical reactions by stochastic automata networks. *Electron. Notes Theor. Comput. Sci.*, 171:197—208, July 2007.

[10] Markus Hegland and Jochen Garcke. On the numerical solution of the chemical master equation with sums of rank one tensors. *ANZIAM Journal*, 52(0), 2011.

[11] Sergey Dolgov and B. Khoromskij. Simultaneous state-time approximation of the chemical master equation using tensor product formats. *Numerical Linear Algebra with Applications*, 22, 03 2015.

[12] Huy D. Vo and Roger B. Sidje. An adaptive solution to the chemical master equation using tensors. *The Journal of Chemical Physics*, 147(4):044102, 2017.