Stochastic Automata Networks and Tensors with Application to Chemical Kinetics

Marie Neubrander*
Advisor: Roger B. Sidje†

Department of Mathematics, University of Alabama
Tuscaloosa, Alabama

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 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. 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. A 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. Applications where biochemical reactions occur are found in a variety of scenarios, including enzyme kinetics and genetics. Tensor-based solutions may have applications in dealing with many other high dimensional data outside of strictly chemical reaction systems.

1 Introduction

In all fields of study, there is an ever increasing need to effectively handle large quantities of data. In particular, models for complex real-world systems frequently require huge data sets that must be stored to allow for efficient mining, analysis, and computations. In this context, one often has to manage *high-dimensional data*—data with a large number of variables being captured; multidimensional arrays, also known as *tensors*, are a useful means of handling such data.

One area of particular interest that deals with high dimensional data is in analyzing biochemical reaction systems; in this work, we model these systems as continuous time Markov processes via the *chemical master equation* (CME), as is discussed for instance in [3, 4, 7, 8, 9]. Such systems consist of a collection of species of molecules and a set of chemical reactions, each with an associated reaction rate; the primary aim is to predict the probability of having certain quantities of each species present at various time points. A particular focus of the literature on the CME is in the creation of the *CME operator* (effectively, a transition matrix) as this problem is large scale, high dimensional, and fundamental to the goal of predicting probabilities.

In 2011, Hegland et al. [3] initiated a method for using tensors in the CME and in the generation of the CME operator; in 2014, Kazeev et. al [4] showed how *tensor train decomposition* can further improve the efficiency of this tensor-based approach. Prior to these developments, Wolf [9] had earlier introduced a CME

*mneubrander@crimson.ua.edu
†roger.b.sidje@ua.edu

method based on *stochastic automata networks*. However, the work [9] did not present its algorithm in a straightforward programmatic form, nor did it dwell on improving efficiency via tensor train compressions as in [4]. Consequently, most of the subsequent attention in the literature has gone to non-SAN approaches.

In this work, we notice the potential for further exploration of SAN-based methods and remodel the algorithm presented in [9] into a more compact algebraic representation. Via computational examples and mathematical proof, we demonstrate the fundamental similarity between Hegland et al.'s operator and Wolf's. This similarity reveals that tensor compressions that were subsequently applied in Kazeev et al. [4] with the representation in Hegland et al. [3] could have been done with the representation in Wolf [9] as well.

The paper first introduces in Section 2 fundamental mathematical tools, including *tensors*, *stochastic automata networks*, and notations for representing chemical reactions; in Section 3, we outline the basic principles of the CME. Section 4 describes in detail two prominent methods used in modeling via the CME, compares results of the two methods via computational examples, and gives a mathematical proof that the similarities seen in the examples hold in all cases—this is the main contribution of the work. Finally, Section 5 presents ideas for generalizing our results and for potential future works.

2 Mathematical Background and Notation

Before beginning modeling reaction systems, we will first go over necessary notations and definitions relating to matrices, tensors, automata networks, and chemical reactions.

2.1 Matrices and Tensors

An N-dimensional (or order-N) tensor is an array $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ where for $n = 1, 2, \ldots, N$, the size of the n^{th} dimension 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 $\mathcal{X}_{i_1, i_2, \ldots, i_N}$—which corresponds to the element $(i_1, i_2, \ldots, i_N)$ of $\mathcal{X}$—for i_n in the n^{th} dimension's indexing range [5]. As shown in Figure 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.

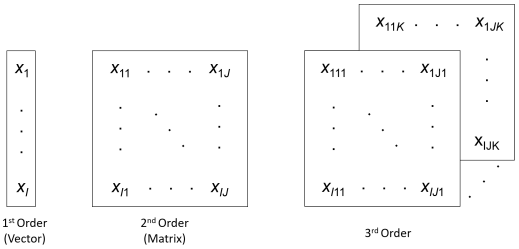


Figure 1: Visualization of 1^{st}, 2^{nd}, and 3^{rd} order tensors in $\mathbb{R}^I$, $\mathbb{R}^{I \times J}$, and $\mathbb{R}^{I \times J \times K}$, respectively.

While these can be easily visualized in 1D, 2D, and 3D as gridpoints as is shown in Figure 2, visualization is harder in higher dimensions, but the concept of gridpoints extends to integer lattices or hyper-rectangles. One additional concept necessary to define for tensors are *slices*, which are given by fixing all indices of a

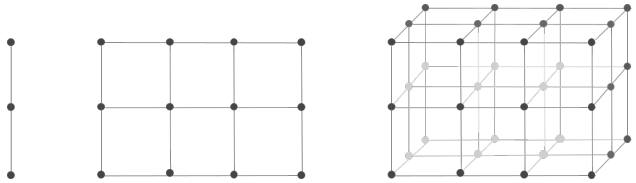


Figure 2: 1D, 2D, and 3D gridpoint visualization.

tensor except for two [5]. As an example, the stacked matrices composing the 3^{rd} order tensor of Figure 1 are slices fixed on the third index.

Basic Operations: Addition, Multiplication, etc.

Throughout this work, we employ several standard matrix operations; the two most relevant are addition and multiplication. Matrix addition is performed on two $I \times J$ matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ where the resultant matrix $\boldsymbol{C}$ has element c_{ij} given by

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

This definition of matrix addition extends naturally to higher order tensors. Standard matrix multiplication between an $I \times J$ matrix $\boldsymbol{A}$ and an $J \times K$ matrix $\boldsymbol{B}$ yields an $I \times K$ matrix $\boldsymbol{C}$ with element c_{ik} given by

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

Extending such matrix multiplication to 3^{rd} or higher order tensors leads to schemas of varying sophistication [5].

Advanced Operations: Kronecker, etc.

Beyond standard matrix multiplication and addition, we can also define several more complex operations, such as the Hadamard, Khatri-Rao, and Kronecker products [5]; of these, the Kronecker product (also known as the tensor product) is the most important to this work. This product of two matrices $\boldsymbol{A}$ (size $I \times J$) and $\boldsymbol{B}$ (size $K \times L$) is denoted $\boldsymbol{A} \otimes \boldsymbol{B}$ and results in a matrix of size $(IK) \times (JL)$. It is defined as

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

which contains each element of $\boldsymbol{A}$ scalar-multiplied by $\boldsymbol{B}$ [5].

2.2 Stochastic Automata Networks

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

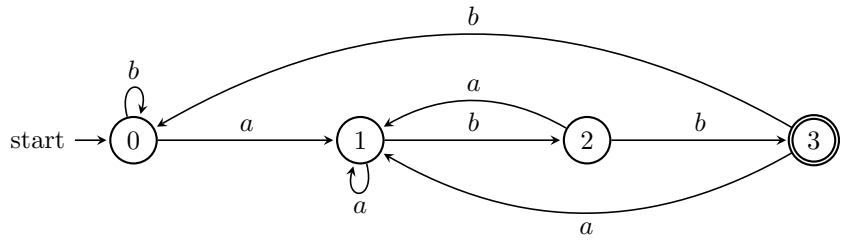
2.2.1 Finite Automata and Regular Expressions

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 [1]. 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 [1]. One useful way of representing DFA is via diagrams or tables of state changes.

2.2.2 Transition Diagrams and Tables

The concept of DFA becomes substantially clearer through an example. For instance, say we wanted to represent the regular expression $(a|b)^{\star}abb$ using DFA (as is presented in [1]). Here, we are taking in strings

of any length of $a's$ and $b's$; 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 3 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 blank entry indicates there is no change, corresponding to a self-loop in the diagram.



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

Figure 3: DFA transition diagram (left) and table (right) for the regular expression $(a|b)^\star abb$.

To see the automata in action, take for instance the input string $aabb$, which results in one possible transition between states given by

$$0 \xrightarrow{a} 1 \xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{b} 3.$$

While these representations may not be realistic for very large-scale examples, they do provide benefits for visualizing and understanding the functioning of systems [1].

2.2.3 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.

2.3 Biochemical Reaction Systems

The final background we will introduce is the language and notation used to represent biochemical reaction systems. Additionally, we will define the *Michaelis-Menten* Reaction System—a simple but important system that will frequently serve as an illustrative example.

2.3.1 General Reaction Systems

When discussing a reaction system, there are several key components, including the species of molecules involved with 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 [4, 9]. First, we define J to be the number of types of molecular species in the reaction system; the species are given by S_j for $j = 1, 2, \ldots, J$. Additionally, we want to describe the quantities of the various species present at some point in time; for this, letting $\mathbb{N}$ be

the set of natural numbers including 0, we define a *state* to be a nonnegative integer vector

$$\boldsymbol{x} := (x_1, x_2, \ldots, x_J) \in \mathbb{N}^J,$$

meaning $\boldsymbol{x}$ is a state in which species S_j has a quantity of x_j molecules. For a system consisting of M possible states, the *state space* is defined to be the set

$$\left\{\boldsymbol{x}_m = (x_1^{(m)}, x_2^{(m)}, \ldots, x_J^{(m)}) \in \mathbb{N}^J\right\}_{m=1}^{M}$$

where the state $\boldsymbol{x}_m$ may be identified simply by its state index m. Transitions between states occur via a number of I chemical reactions R_i, for $i \in \mathbb{N}$ from 1 to I. The equations for the reactions are given by

$$R_i : s_{i1}S_1 + s_{i2}S_2 + \ldots + s_{iJ}S_J \xrightarrow{c_i} s_{i,J+1}S_1 + s_{i,J+2}S_2 + \ldots + s_{i,2J}S_J, \tag{2.1}$$

where the positive $c_i \in \mathbb{R}$ is the rate of reaction R_i, assumed to be constant here, although it can be variable in cases not considered here (such as a change of temperature or volume). The values $s_{i1}, \ldots s_{i,2J}$ are stoichiometric coefficients indicating how much of each species is involved in either side of the reaction; we assume that the molecules appear in the same order on both sides of the equation. The coefficient of species S_j on the left-hand side of reaction R_i is given by s_{ij}; its coefficient on the right-hand side is given by $s_{i,J+j}$. A convenient way of representing this information is via an $I \times 2J$ *stoichiometry matrix* in which entry (i, j) represents the corresponding coefficient s_{ij}; it has the form

$$\left[\begin{array}{ccc|ccc} s_{11} & \ldots & s_{1J} & s_{1,J+1} & \ldots & s_{1,2J} \\ \vdots & & \vdots & \vdots & & \vdots \\ s_{I1} & \ldots & s_{IJ} & s_{I,J+1} & \ldots & s_{I,2J} \end{array}\right] \in \mathbb{N}^{I \times 2J}. \tag{2.2}$$

Using a similar notation, we also define an $I \times J$ *stoichiometry change matrix* that indicates how much a given reaction changes the quantities of each species. Entry (i, j) represents how much reaction R_i changes molecule type S_j and is given by $s_{i,J+j} - s_{ij}$; this matrix has the form

$$\begin{bmatrix} s_{1,J+1} - s_{11} & \ldots & s_{1,2J} - s_{1J} \\ \vdots & & \vdots \\ s_{I,J+1} - s_{I1} & \ldots & s_{I,2J} - s_{IJ} \end{bmatrix} \in \mathbb{Z}^{I \times J}. \tag{2.3}$$

Based on the stoichiometric coefficients, we define the following three types of species [9]:

- *Reactants*: Have a non-zero coefficient on the left side of the reaction (2.1); the set of reactant species for R_i are notated by $\mathbf{REA}(i)$.
- *Products*: Have a non-zero coefficient on the right side of the reaction (2.1); the set of product species for R_i are notated by $\mathbf{PRO}(i)$.
- *Catalysts*: Have non-zero coefficients on both the right and left-hand sides of the reaction (2.1) and a zero value in the change matrix (2.3); the set of catalyst species for R_i are given by $\mathbf{CAT}(i) = \mathbf{REA}(i) \cap \mathbf{PRO}(i)$. [1]

Finally, we want to be able to analyze the probability of being in a particular state at a given time t. To do so, we define a probability vector $\boldsymbol{p}(t)$. This column vector is given by

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

in which each entry p_m represents the probability of being in state $\boldsymbol{x}_m$ at time t.

[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.

2.3.2 Application Example: Michaelis-Menten Reaction System

In the above section, we defined the general form of representing reaction systems. Here, we demonstrate a specific reaction system—specifically, the Michaelis-Menten system—which will be used in several illustrative examples.

In the Michaelis-Menten reactions, a substrate (S) is catalyzed by an enzyme (E) to form an intermediate substrate-enzyme complex (ES) and a product (P) [7]. Throughout our 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. This system models three reactions defined as

$$\begin{cases} R_1 & : \quad S + E \xrightarrow{c_1} ES \\ R_2 & : \quad S + E \xleftarrow{c_2} ES \\ R_3 & : \quad ES \xrightarrow{c_3} P + E \end{cases} . \tag{2.4}$$

From here, we can write the reactions in their full forms to see all stoichiometric coefficients and the stoichiometry matrix (2.2), which in this example turns out to be

$$\begin{array}{cccccccccc} S & E & E & P & \rightarrow & S & E & ES & P & \\ \left[\begin{matrix} 1 \\ 0 \\ 0 \end{matrix}\right. & \begin{matrix} 1 \\ 0 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 1 \\ 1 \end{matrix} & \left.\begin{matrix} 0 \\ 0 \\ 0 \end{matrix}\right] & & \left[\begin{matrix} 0 \\ 1 \\ 0 \end{matrix}\right. & \begin{matrix} 0 \\ 1 \\ 1 \end{matrix} & \begin{matrix} 1 \\ 0 \\ 0 \end{matrix} & \left.\begin{matrix} 0 \\ 0 \\ 1 \end{matrix}\right] & \begin{matrix} S + E \xrightarrow{c_1} ES \\ ES \xrightarrow{c_2} S + E \\ ES \xrightarrow{c_3} P + E \end{matrix} \end{array}$$

Additionally, the entries of the stoichiometry change matrix (2.3) work out in this example to be

$$\begin{array}{cccc} S & E & ES & P & \\ \left[\begin{matrix} -1 \\ 1 \\ 0 \end{matrix}\right. & \begin{matrix} -1 \\ 1 \\ 1 \end{matrix} & \begin{matrix} 1 \\ -1 \\ -1 \end{matrix} & \left.\begin{matrix} 0 \\ 0 \\ 1 \end{matrix}\right] & \begin{matrix} S + E \xrightarrow{c_1} ES \\ ES \xrightarrow{c_2} S + E \\ ES \xrightarrow{c_3} P + E \end{matrix} \end{array}$$

In the coming sections, we will elaborate on how one can use this form of representation to analyze and make predictions on these systems.

3 The Chemical Master Equation (CME)

In the above sections, we defined how we can mathematically represent biochemical reaction systems; here, we briefly outline how to set up the *chemical master equation* to model these systems. Particularly, we are interested in predicting the probability of being in a particular state after a given amount of time.

3.1 Overview

To set up the chemical master equation, we represent a reaction system as a continuous time Markov process in which the future state is dependent on the current state [9, 7, 4, 8]. In addition to the previous components, we also use a *CME operator* (effectively a transition matrix) $\boldsymbol{A}$ where element $a_{mk}, m \neq k$ is the propensity of changing to state m when the system is currently in state k to model these systems. Starting with an initial probability vector $\boldsymbol{p}(0)$, the probabilities of being in each state at each period of time are found by solving the system of differential equations [4]

$$\begin{cases} \dfrac{d}{dt}\boldsymbol{p} & = \boldsymbol{A} \cdot \boldsymbol{p} \\ \boldsymbol{p}(0) & = \boldsymbol{p}_0 \end{cases} \tag{3.1}$$

which has exact solution

$$\boldsymbol{p}(t) = \exp(t\boldsymbol{A})\boldsymbol{p}(0). \tag{3.2}$$

One important question that arises when performing this modeling is *what are the possible states?* First, we will focus on a finite set of states that we know are all possible reachable states from a given starting state; eventually, we will discuss the set of states as all possible combinations of molecules for a given quantity of molecules.

3.2 Application Example: Michaelis-Menten

To clearly illustrate the functionality of the chemical master equation, let us examine a small example involving the following five states and transition matrix corresponding to the reaction system (2.4):

$$\begin{aligned}
\boldsymbol{x}_1 &= [2,1,0,0] \\
\boldsymbol{x}_2 &= [1,0,1,0] \\
\boldsymbol{x}_3 &= [1,1,0,1] \\
\boldsymbol{x}_4 &= [0,0,1,1] \\
\boldsymbol{x}_5 &= [0,1,0,2]
\end{aligned} \qquad \boldsymbol{A} = \begin{bmatrix} -2 & 1 & 0 & 0 & 0 \\ 2 & -1.1 & 0 & 0 & 0 \\ 0 & 0.1 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1.1 & 0 \\ 0 & 0 & 0 & 0.1 & 0 \end{bmatrix} \tag{3.3}$$

The transition matrix and list of possible states are calculated using the initial state (here, $\boldsymbol{x}_1$), reactions, and propensities; in this example, we will take the states and matrix as given. Later, we will generalize the concept of obtaining the state space and a transition matrix.

To show the transitions between states, we can use a transition diagram as was discussed in Section 2.2.2—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. Note that as each of these reactions has a different probability of occurring, this is a stochastic automata model. This is illustrated in Figure 4; here, we assume we start in state $\boldsymbol{x}_1$. Furthermore, 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. We will see in Section 4.2 how this can be conveniently formalized with tensors.

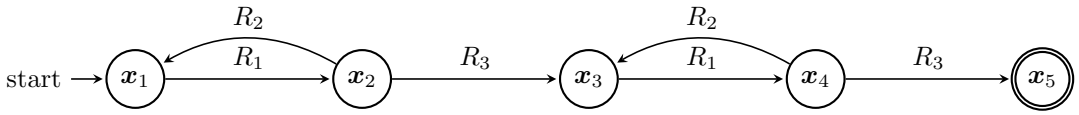


Figure 4: Stochastic Automata: Michaelis-Menten 5-State Example

Now, we wish to predict the probability of being in a particular state at time t. Assuming state one to be the starting state, there is a 100% chance of being in state one at time zero, so

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

Using Equation (3.1), we can now calculate the probability vector for some time t; for example, with $t = 10$ and $t = 100$, the resulting probability vectors are

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

In this example, at time $t = 10$, there is the highest probability of the system being in state 2. However, by $t = 100$, it is most likely that the system will be in state 5. Figure 5 further illuminates the change in probabilities over time, plotting the probabilities of being in each state (y-axis) from times 0 to 100 in intervals of 20 (x-axis).

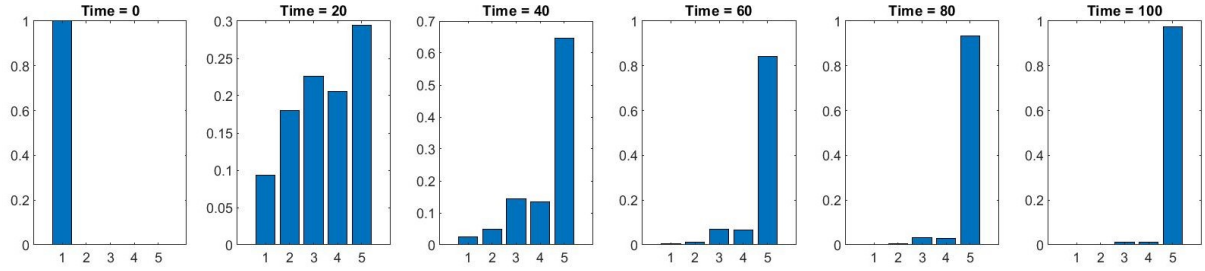


Figure 5: Probabilities of being in states 1-5 at times 0, 20, 40, 60, 80, and 100.

Recall that in this example, we both took the transition matrix as given and assumed that we were dealing only with a finite and known list of reachable states. Neither of these assumptions, however, are realistic to what the real modeling problem looks like. In fact, our most fundamental question arises in determining how we generate the transition matrix, which will from this point forwards be generalized and referred to as the *CME operator* [3, 4]. Furthermore, it is known that, without an imposed bound on the molecules' quantities, the state space can be arbitrarily large. We further introduce and explore these problems in the following sections.

3.3 Truncated State Space

The goal of the CME operator is to capture the transition from any one state to any other. However, there could theoretically be a 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 [4]. Each type of molecule S_j is given a maximum amount $n_j \in \mathbb{N}$ that may be present in any given state; i.e. each S_j can have anywhere from 0 to n_j molecules, so the state space consists of $M = (n_1 + 1)(n_2 + 1)\dots(n_J + 1)$ possible states. The state space is then given by the lattice or hyper-rectangle [9]

$$\{(x_1, x_2, \dots, x_J) \in \mathbb{N}^J \mid 0 \leq x_j \leq n_j,\ j \in \{0, 1, \dots, J\}\}.$$

This results in $\boldsymbol{A} \in \mathbb{R}^{M \times M}$ and $\boldsymbol{p} \in \mathbb{R}^M$ where M is extremely large in realistic applications. For example, if $n_j \approx 100$ and $J = 5$ species, then M is already 10^{10}. For the rest of the paper, the term state space will refer to the truncated state space described in this section, unless stated otherwise.

4 The Chemical Master Equation via Tensors

As we just saw, with $M, J \gg 1$, the problem of modeling biochemical reaction systems is high dimensional, which presents an opportunity for incorporating higher-order tensors [3, 4]. Particularly, notice that instead of enumerating the states and writing $\boldsymbol{A}$ as a matrix, one may instead write the CME operator as an order $2J$ tensor

$$\boldsymbol{\mathcal{A}} \in \mathbb{R}^{(n_1+1)\times\cdots\times(n_J+1)\times(n_1+1)\times\cdots\times(n_J+1)},$$

where the entry $\mathcal{A}_{x_1,\ \dots,\ x_J,\ x_{J+1},\ \dots,\ x_{2J}}$ represents the propensity of transitioning from state $(x_{J+1}, \dots, x_{2J})$ to state $(x_1, \dots, x_J)$, such that each molecule S_j changes from quantity x_{J+j} to x_j.

4.1 Classic Approach

In 2011, Hegland et al. [3] 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 [4] presented the incorporation of *tensor train decompositions* to the computational procedure of [3] while Dolgov and Khoromskij [2] 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 very quickly such 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 [8].

4.1.1 Algorithm

We give 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. [3]. We use the algorithmic version and notation in Kazeev et al. [4] where the resultant CME operator matrix $\boldsymbol{A}$ is generated by

$$\boldsymbol{A} = \sum_{i=1}^{I} (\boldsymbol{S}_{\eta^i} - \boldsymbol{I\!I}) * \boldsymbol{M}_{\omega^i},$$

in which each $\boldsymbol{I\!I}$ indicates the identity operator and each reaction R_i yields corresponding $\boldsymbol{S}_{\eta^i}$ and $\boldsymbol{M}_{\omega^i}$ matrices that we will detail below. A reaction's $\boldsymbol{M}_{\omega^i}$ is a diagonal matrix associated to a vector $\boldsymbol{\omega}^i$ of its diagonal entries. For each reaction R_i and species S_j, define the intermediate vector composed of binomial coefficients

$$\boldsymbol{\omega}_j^i = \left(\binom{0}{s_{ij}}, \cdots, \binom{n_j}{s_{ij}} \right). \tag{4.1}$$

Using these, $\boldsymbol{\omega}^i$ is given by

$$\boldsymbol{\omega}^i = c_i \bigotimes_{j=1}^{J} \boldsymbol{\omega}_j^i.$$

Similarly, a reaction's matrix $\boldsymbol{S}_{\eta^i}$ is defined by first creating matrices $\boldsymbol{S}_{\eta_j^i}$, $j = 1, \ldots, J$ (one for each species); these $\boldsymbol{S}_{\eta_j^i}$ matrices are shifted identity matrices in which the amount and direction shifted is dependent on the corresponding values of n_j and the change matrix at (i, j). The formulation of the matrix is [4]

$$\boldsymbol{S}_{\eta^i} = \bigotimes_{j=1}^{J} \boldsymbol{S}_{\eta_j^i}.$$

In [4], the quantized tensor train approximations of the $\boldsymbol{\omega}_j^i$ are formed before the Kronecker products are computed; although this was a crucial aspect of Kazeev et al. [4], this highly beneficial tensor compression step is noted as a comment in Algorithm 3 but is not applied in Algorithm 1 because our own focus is on the formation of the operator via Kronecker products.

Further meanings behind these matrices and how they are computed are demonstrated through Algorithms 1, 2, and 3 where the operator $\text{diag}(\boldsymbol{v})$ forms the diagonal matrix with the elements of the vector $\boldsymbol{v}$ on its diagonal. Refer to [4] for a more detailed explanation of this presentation of the approach[2]; refer to [3] for a more detailed explanation of the original presentation of the approach.

[2]The indices with i from 1 to I correspond to indices s from 1 to R in [4]. This modification simplifies the comparison with the method presented in Section 4.2.

Algorithm 1: Classic Approach [4, Alg. 1]

Data: $\boldsymbol{stoichmatrix}$, $\boldsymbol{nsize}$, $\boldsymbol{c}$
// $\boldsymbol{stoichmatrix}$ is the stoichiometry matrix; refer to Equation 2.2
// $\boldsymbol{nsize} = (n_1 + 1, \ldots, n_J + 1)$; j^{th} element is $n_j + 1$
// $\boldsymbol{c} = (c_1, \ldots, c_I)$; vector of reaction rate constants with i^{th} element = c_i.
Result: $\boldsymbol{A}$

$I \leftarrow$ number of reactions // #rows of $\boldsymbol{stoichmatrix}$
$J \leftarrow$ number of species // #cols of $\boldsymbol{stoichmatrix}$ / 2
$\boldsymbol{changes} \leftarrow$ stoichiometry change matrix // create from $\boldsymbol{stoichmatrix}$; refer to Equation (2.3)
$M \leftarrow$ number of states // product of elements of $\boldsymbol{nsize}$
$\boldsymbol{I\!I} \leftarrow$ identity matrix of size M
$\boldsymbol{A} \leftarrow$ zero matrix of size M
for $i \leftarrow 1{:}I$ **do**
    for $j \leftarrow 1{:}J$ **do**
        $\boldsymbol{S}_{\eta_j^i} \leftarrow$ shifted identity matrix resulting from **Algorithm 2**
        $\boldsymbol{\omega}_j^i \leftarrow$ vector resulting from **Algorithm 3**
        // Do kronecker products to generate S_{η^i} and M_{ω^i}
        if $j = 1$ **then**
            $\boldsymbol{S}_{\eta^i} \leftarrow \boldsymbol{S}_{\eta_j^i}$ // matrix
            $\boldsymbol{\omega}^i \leftarrow c_i * \boldsymbol{\omega}_j^i$ // vector
        else
            $\boldsymbol{S}_{\eta^i} \leftarrow \boldsymbol{S}_{\eta^i} \otimes \boldsymbol{S}_{\eta_j^i}$ // matrix
            $\boldsymbol{\omega}^i \leftarrow \boldsymbol{\omega}^i \otimes \boldsymbol{\omega}_j^i$ // vector
        end
    end
    $\boldsymbol{M}_{\omega^i} \leftarrow \mathrm{diag}(\boldsymbol{\omega}^i)$
    $\boldsymbol{A} \leftarrow \boldsymbol{A} + (\boldsymbol{S}_{\eta^i} - \boldsymbol{I\!I}) * \boldsymbol{M}_{\omega^i}$
end

Algorithm 2: Create $\boldsymbol{S}_{\eta_j^i}$

Data: $\boldsymbol{nsize}(j)$, $\boldsymbol{changes}(i,j)$
// Input data values come from Algorithm 1
Result: $\boldsymbol{S}_{\eta_j^i}$

$t \leftarrow |\boldsymbol{changes}(i,j)|$
$\boldsymbol{S}_{\eta_j^i} \leftarrow$ zero matrix of size $\boldsymbol{nsize}(j)$
Replace $\boldsymbol{S}_{\eta_j^i}(1 : \boldsymbol{nsize}(j) - t,\ 1 + t : \boldsymbol{nsize}(j))$ with identity matrix of size t // as in Equation (4.11a)
if $\boldsymbol{changes}(i,j) > 0$ **then**
    $\boldsymbol{S}_{\eta_j^i} \leftarrow$ transpose of $\boldsymbol{S}_{\eta_j^i}$ // as in Equation (4.11b)
end

Algorithm 3: Create $\boldsymbol{\omega}_j^i$

Data: $\boldsymbol{nsize}(j)$, s_{ij}
// Input data values come from Algorithm 1
Result: $\boldsymbol{\omega}_j^i$

$n_j \leftarrow \boldsymbol{nsize}(j) - 1$
$\boldsymbol{\omega}_j^i \leftarrow \left(\binom{0}{s_{ij}}, \ldots, \binom{n_j}{s_{ij}}\right)$ // vector components are binomial coefficients, refer to Equation (4.1)
// Kazeev et al. [4] achieve their compression by applying a QTT_Approx to $\boldsymbol{\omega}_j^i$

4.1.2 Application Example: Michaelis-Menten

In this example, we expand on that given in Section 3.2; 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 $n_2 = n_3 = 1$, so the size of the state space is given by $3 \times 2 \times 2 \times 3 = 36$; the resulting matricized CME operator $\boldsymbol{A}$ will have a size of 36×36. When using Algorithm 1, the generated matrix is shown in Figure 6; a blank indicates that the corresponding location has a zero value, while non-zero values are given in their positions. When looking at this matrix, note that it is slightly different from the matrix given in Equation (3.3). The entries of the latter can be found in the former, though not necessarily with the same state number; this is expected, as that small state space is a subset of this one.

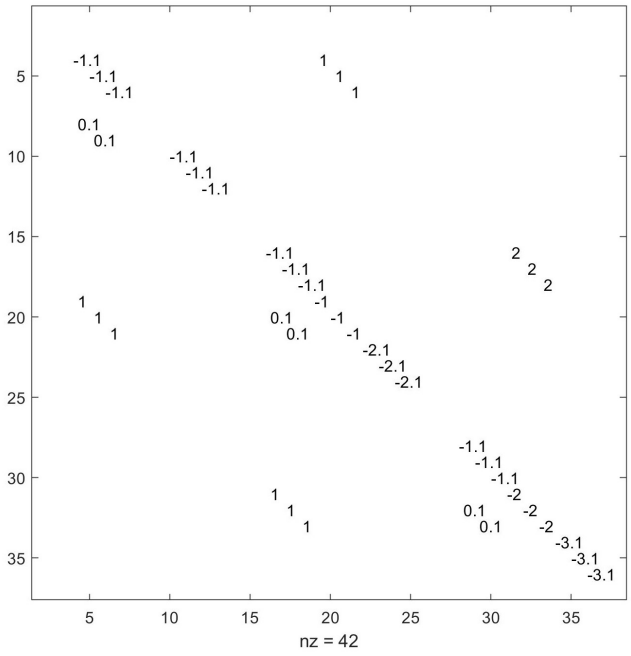


Figure 6: The shape and 42 non-zero values of the CME operator generated in Section 4.1.2. Columns sum to zero, except on boundary states where probability leaks occur (more on this in Section 4.3.1).

4.2 SAN Approach

Prior to and separate from the approach summarized in Section 4.1 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 [3] and made more efficient by [4], which is a striking discovery that has not been noted until now and is the main contribution of this work.

4.2.1 Modeling Method

This section presents the fundamental components of the methods from [9] (with slight modifications to some notations).

First, to go from state $\boldsymbol{x}$ to state $\boldsymbol{x}'$, Wolf [9] defines a function $\text{next}_i(\boldsymbol{x}) = \boldsymbol{x}'$ with $\boldsymbol{x}'$ given by

$$x'_j = \begin{cases} x_j & \text{if } S_j \in \mathbf{CAT}(i) \text{ or } S_j \notin \mathbf{REA}(i) \cup \mathbf{PRO}(i) \\ x_j - s_{ij} & \text{if } S_j \in \mathbf{REA}(i) \setminus \mathbf{CAT}(i) \\ x_j + s_{i,j+J} & \text{if } S_j \in \mathbf{PRO}(i) \setminus \mathbf{CAT}(i) \end{cases}$$

There is additionally the function

$$\text{rate}_i(\boldsymbol{x}) = \begin{cases} c_i \cdot \prod_{S_h \in \mathbf{REA}(i)} \binom{x_h}{s_{ih}} & \text{if } \text{next}_i(\boldsymbol{x}) \neq \boldsymbol{x} \\ 0 & \text{else} \end{cases}$$

which determines the transition rate of a state $\boldsymbol{x}$ acted on by reaction R_i and is used to define a state's *exit rate* as $\text{rate}(\boldsymbol{x}) = \text{rate}_1(\boldsymbol{x}) + \ldots + \text{rate}_I(\boldsymbol{x})$. It is from these that [9] defines the transition rate matrix $\boldsymbol{Q} \in \mathbb{R}^{M \times M}$ with entries

$$Q(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} -\text{rate}(\boldsymbol{x}) & \text{if } \boldsymbol{x} = \boldsymbol{x}' \\ \sum\limits_{i:\text{next}_i(\boldsymbol{x})=\boldsymbol{x}'} \text{rate}_i(\boldsymbol{x}) & \text{if } \boldsymbol{x}' \neq \boldsymbol{x} \\ 0 & \text{else} \end{cases}$$

Here, it is important to note that $\boldsymbol{Q}$ is analogous to the transpose of the CME operator $\boldsymbol{A}$ from Equations (3.1) and (3.2). The work in [9] was to show how to generate $\boldsymbol{Q}$ using a tensor representation through the Kronecker product. To accomplish this, [9] first defines the vector

$$\mathbf{dep}_j^i = \left(\binom{0}{s_{ij}}, \ldots, \binom{n_j}{s_{ij}} \right) \in \mathbb{N}^{(n_j+1)} \tag{4.2}$$

for each reactant $S_j \in \mathbf{REA}(i)$. For species S_j that are not reactants of R_i, [9] instead defines

$$\mathbf{ind}_j^i = (1, \ldots, 1) \in \mathbb{N}^{(n_j+1)}.$$

Using these vectors, [9] defines $(n_j + 1) \times (n_j + 1)$ matrices $\mathbf{Dep}_j^i(d)$ and $\mathbf{Ind}_j^i(d)$—these shift the corresponding **dep** and **ind** vectors to a d^{th} diagonal. For k, l such that $(k + d) = l$ where $1 \leq k,\ l \leq (n_j + 1),\ -n_j \leq d \leq n_j$, the element at position (k, l) of $\mathbf{Dep}_j^i(d)$ is equal to the k^{th} element of $\mathbf{dep}_j^i$ and the element at position (k, l) of $\mathbf{Ind}_j^i(d)$ equals 1. All other entries are set to zero. Using these definitions, [9] defines a matrix $\boldsymbol{E}_j^{(i)}$ for each reaction R_i and species S_j given by

$$\boldsymbol{E}_j^{(i)} = \begin{cases} \mathbf{Ind}_j^i(0) & \text{if } S_j \notin \mathbf{REA}(i) \cup \mathbf{PRO}(i) \\ \mathbf{Dep}_j^i(0) & \text{if } S_j \in \mathbf{CAT}(i) \\ \mathbf{Dep}_j^i(-s_{ij}) & \text{if } S_j \in \mathbf{REA}(i) \setminus \mathbf{CAT}(i) \\ \mathbf{Ind}_j^i(s_{i,j+J}) & \text{if } S_j \in \mathbf{PRO}(i) \setminus \mathbf{CAT}(i) \end{cases} \tag{4.3}$$

and a corresponding diagonal matrix $\boldsymbol{D}_j^{(i)}$ given by

$$\boldsymbol{D}_j^{(i)} = \text{diag}(\boldsymbol{E}_j^{(i)} \mathbb{1}) \tag{4.4}$$

where $\mathbb{1}$ is a column vector of entirely ones. Wolf [9] finally generates $\boldsymbol{Q}$ as

$$\boldsymbol{Q} = \sum_{i=1}^{I} c_i \left(\bigotimes_{j=1}^{J} \boldsymbol{E}_j^{(i)} - \bigotimes_{j=1}^{j} \boldsymbol{D}_j^{(i)} \right). \tag{4.5}$$

Note that 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 [4]; we will comment more on this in Section 4.3.3.

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 $\boldsymbol{E}_j^{(i)}$ in Equation (4.3).

Here, we remodel these conditional statements into a more compact algebraic representation that we use to implement an automated algorithm in MATLAB to generate $\boldsymbol{Q}$. First, we examine the values for the diagonal shift d in $\mathbf{Dep}_j^i(d)$ and $\mathbf{Ind}_j^i(d)$ used in Equation (4.3) in the four different cases.

- If S_j is not involved in reaction R_i, so $S_j \notin \mathbf{REA}(i) \cup \mathbf{PRO}(i)$, then $s_{ij} = s_{i,J+j} = 0$. Thus, $s_{i,J+j} - s_{ij} = 0$.
- If S_j is a catalyst in reaction R_i, so $S_j \in \mathbf{CAT}(i)$, then $s_{ij} = s_{i,J+j}$. Thus, $s_{i,J+j} - s_{ij} = 0$.
- If S_j is a reactant (but not catalyst) in reaction R_i, so $S_j \in \mathbf{REA}(i) \setminus \mathbf{CAT}(i)$, then $s_{i,J+j} = 0$. Thus, $s_{i,J+j} - s_{ij} = -s_{ij}$.
- If S_j is a product (but not catalyst) in reaction R_i, so $S_j \in \mathbf{PRO}(i) \setminus \mathbf{CAT}(i)$, then $s_{ij} = 0$. Thus, $s_{i,J+j} - s_{ij} = s_{i,J+j}$.

For all cases, the shift value is then given by $s_{i,J+j} - s_{ij}$. Now, we examine the usage of **Dep** versus **Ind** matrices. Note that if $S_j \notin \mathbf{REA}(i) \cup \mathbf{PRO}(i)$ or $S_j \in \mathbf{PRO}(i) \setminus \mathbf{CAT}(i)$ then $s_{ij} = 0$; for any $n \in \mathbb{Z},\ n \geq 0,\ \binom{n}{0} = 1$. Thus, for such S_j the entries of $\mathbf{dep}_j^i$ would be all ones, which is the definition of $\mathbf{ind}_j^i$, so we can re-write the formulation using only **Dep** matrices and corresponding **dep** vectors. Using these observations, we re-write the definition of matrix $\boldsymbol{E}_j^{(i)}$ in Equation (4.3) as

$$\boldsymbol{E}_j^{(i)} = \mathbf{Dep}_j^i\,(s_{i,J+j} - s_{ij}). \tag{4.6}$$

Algorithms 4, 5, and 6 are pseudo-codes, based on this re-formulation of the work in [9].

Algorithm 4: SAN Approach

Data: ***stoichmatrix***, ***nsize***, c

// ***stoichmatrix*** is the stoichiometry matrix; refer to Equation 2.2

// ***nsize*** $= (n_1 + 1, \ldots, n_J + 1)$; j^{th} element ***nsize***(j) is $n_j + 1$

// $\boldsymbol{c} = (c_1, \ldots, c_I)$; vector of reaction rate constants with i^{th} element = c_i.

Result: $\boldsymbol{Q}$

```
I ← number of reactions  // #rows of stoichmatrix
J ← number of species  // #cols of stoichmatrix / 2
changes ← stoichiometry change matrix  // create from stoichmatrix; refer to Equation 2.3
M ← number of states  // product of elements of nsize
Q ← zero matrix of size M
for i ← 1: I do
    for j ← 1: J do
        create dep_j^i using Algorithm 5
        create E_j^(i) using Algorithm 6
        create D_j^(i) using Equation (4.4)
        // Do kronecker products to get E^(i) = ⊗_{j=1}^J E_j^(i) and D^(i) = ⊗_{j=1}^J D_j^(i)
        if j = 1 then
            E^(i) ← E_j^(i);      D^(i) ← D_j^(i)
        else
            E^(i) ← E^(i) ⊗ E_j^(i);      D^(i) ← D^(i) ⊗ D_j^(i)
        end
    end
    Q ← Q + c_i * (E^(i) − D^(i))
end
```

Algorithm 5: Create $\mathbf{dep}_j^i$

Data: $\boldsymbol{nsize}(j),\ s_{ij}$
// Input data values come from Algorithm 4
Result: $\mathbf{dep}_j^i$

$n_j \leftarrow \boldsymbol{nsize}(j) - 1$
$\mathbf{dep}_j^i \leftarrow \left(\binom{0}{s_{ij}}, \ldots, \binom{n_j}{s_{ij}}\right)$ // vector components are binomial coefficients; refer to Equation (4.2)

Algorithm 6: Create $\boldsymbol{E}_j^{(i)}$

Data: $\mathbf{dep}_j^i,\ \boldsymbol{changes}(i,j)$
// Input data values come from Algorithm 4
Result: $\boldsymbol{E}_j^{(i)}$

$l \leftarrow$ length of $\mathbf{dep}_j^i$ // is equal to $n_j + 1$
$t \leftarrow |\boldsymbol{changes}(i,j)|$
$\boldsymbol{E}_j^{(i)} \leftarrow$ zero matrix of size l
if $\boldsymbol{changes}(i,j) > 0$ **then**
    Replace $\boldsymbol{E}_j^{(i)}(1 : l-t\ ,\ 1+t : l)$ with $\mathrm{diag}(\mathbf{dep}_j^i(1 : l-t))$ // as in Equation (4.10a)
else
    Replace $\boldsymbol{E}_j^{(i)}(1+t : l\ ,\ 1 : l-t)$ with $\mathrm{diag}(\mathbf{dep}_j^i(1+t : l))$ // as in Equation (4.10b)
end

4.2.3 Application Example: Michaelis-Menten

We use the method from Section 4.2.2 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 7.

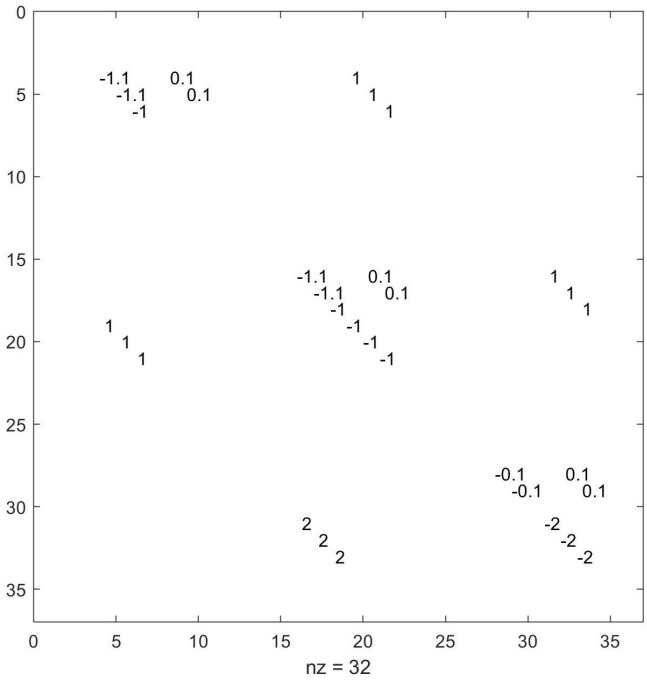


Figure 7: The shape and 32 non-zero values of the CME operator from Section 4.2.3. Rows sum to zero.

It is immediately clear that the matrices in Figures 6 and 7 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 Comparison of Methodologies

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 (see also Section 3.2), 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, the resulting probability vectors may sum to a value less than 1—that is, we see 'probability leaks'.

4.3.1 Probability Leaks

The source of the probability leaks stems from truncating the state space. To clearly see why this is the case, say, for instance, that a Michaelis-Menten system had starting state $[2, 1, 1, 2]$—it would then be possible to transition to state $[1, 0, 2, 2]$. 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 is not found in the state space, and thus the propensity to transition to it from the starting state 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 (4.5), through the creation and subtraction of the diagonal $\boldsymbol{D}_j^{(i)}$ matrices, ensures as in Figure 7 that the rows 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.

4.3.2 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. Here, we indeed prove that the off-diagonal elements resulting from the classic method [3, 4] are identical to the transpose of those resulting from the SAN method [9].

Recall that each algorithm is performed iteratively—for each reaction, they generate several matrices and sum them together—and that the classic methodology's algorithm is summarized by the equation

$$\boldsymbol{A} = \sum_{i=1}^{I} (\boldsymbol{S}_{\eta^i} - \boldsymbol{I\!I}) * \boldsymbol{M}_{\omega^i}$$

where $\boldsymbol{A}$ is the CME operator, with

$$\boldsymbol{S}_{\eta^i} = \boldsymbol{S}_{\eta_1^i} \otimes \cdots \otimes \boldsymbol{S}_{\eta_J^i}, \qquad \boldsymbol{M}_{\omega^i} = \operatorname{diag}(c_i * \boldsymbol{\omega}_1^i \otimes \cdots \otimes \boldsymbol{\omega}_J^i),$$

as described in Section 4.1.1 and shown in Algorithms 1, 2, and 3. The SAN methodology is summarized by

$$\boldsymbol{Q} = \sum_{i=1}^{I} c_i \left(\bigotimes_{j=1}^{J} \boldsymbol{E}_j^{(i)} - \bigotimes_{j=1}^{J} \boldsymbol{D}_j^{(i)} \right),$$

where $\boldsymbol{Q}$ is analagous to the CME operator in transposed form, as is described in Section 4.2. Since $\boldsymbol{M}_{\omega^i}$ and $\bigotimes_{j=1}^{J} \boldsymbol{D}_j^{(i)}$ are diagonal matrices, demonstrating the equality of the off-diagonal elements of $\boldsymbol{A}$ and $\boldsymbol{Q}^T$ can be reduced to demonstrating that

$$\sum_{i=1}^{I} \boldsymbol{S}_{\eta^i} * \boldsymbol{M}_{\omega^i} = \left(\sum_{i=1}^{I} c_i \bigotimes_{j=1}^{J} \boldsymbol{E}_j^{(i)} \right)^T = \sum_{i=1}^{I} \left(c_i \bigotimes_{j=1}^{J} \boldsymbol{E}_j^{(i)} \right)^T, \tag{4.7}$$

and so, if for each $i \in [1, I]$ we have

$$\boldsymbol{S}_{\eta^i} * \boldsymbol{M}_{\omega^i} = \left(c_i \bigotimes_{j=1}^{J} \boldsymbol{E}_j^{(i)} \right)^T ,$$

then Equation (4.7) will hold true. Thus, we aim to show in expanded form that

$$(\boldsymbol{S}_{\eta_1^i} \otimes \cdots \otimes \boldsymbol{S}_{\eta_J^i}) * \mathrm{diag}(c_i * \boldsymbol{\omega}_1^i \otimes \cdots \otimes \boldsymbol{\omega}_J^i) = (c_i * \boldsymbol{E}_1^{(i)} \otimes \cdots \otimes \boldsymbol{E}_J^{(i)})^T .$$

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

$$(\boldsymbol{S}_{\eta_1^i} * \mathrm{diag}(\boldsymbol{\omega}_1^i)) \otimes \cdots \otimes (\boldsymbol{S}_{\eta_J^i} * \mathrm{diag}(\boldsymbol{\omega}_J^i)) = (\boldsymbol{E}_1^{(i)})^T \otimes \cdots \otimes (\boldsymbol{E}_J^{(i)})^T .$$

Thus, the proof is complete if we show that

$$\boldsymbol{S}_{\eta_j^i} * \mathrm{diag}(\boldsymbol{\omega}_j^i) = (\boldsymbol{E}_j^{(i)})^T \tag{4.8}$$

for any reaction R_i and species S_j. To prove Equation (4.8), first note that, as was mentioned previously, the vector $\mathbf{dep}_j^i$ for reaction R_i and molecules S_j used in creating $\boldsymbol{E}_j^{(i)}$ is equivalent to $\boldsymbol{\omega}_j^i$ (refer to Algorithms 3, 5); we have

$$\mathbf{dep}_j^i = \boldsymbol{\omega}_j^i = \left(\binom{0}{s_{ij}}, \cdots, \binom{n_j}{s_{ij}} \right). \tag{4.9}$$

Next, note that $\boldsymbol{E}_j^{(i)} \in \mathbb{R}^{(n_j+1)\times(n_j+1)}$ is given by

$$\begin{cases} \begin{matrix} \\ n_j + 1 - t \end{matrix} \rightarrow \begin{matrix} 1+t \\ \downarrow \end{matrix} \begin{bmatrix} 0 & \cdots & (\mathbf{dep}_j^i)_1 & & \\ & \cdot & & \ddots & \\ & & \cdot & & (\mathbf{dep}_j^i)_{n_j+1-t} \\ & & & \cdot & \vdots \\ & & & & 0 \end{bmatrix} , \; S_j \in \mathbf{PRO}(i) & (4.10a) \\ \begin{matrix} \\ 1 + t \end{matrix} \rightarrow \begin{matrix} n_j + 1 - t \\ \downarrow \end{matrix} \begin{bmatrix} 0 & & & & \\ \vdots & \cdot & & & \\ (\mathbf{dep}_j^i)_{1+t} & & \cdot & & \\ & \ddots & & \cdot & \\ & & (\mathbf{dep}_j^i)_{n_j+1} & \cdots & 0 \end{bmatrix} , \; S_j \notin \mathbf{PRO}(i) & (4.10b) \end{cases} ,$$

where, consistent with the notation in Algorithms 2 and 6, t refers to the absolute value of entry (i, j) of the change matrix. Now, note that $\boldsymbol{S}_{\eta_j^i} \in \mathbb{R}^{(n_j+1)\times(n_j+1)}$ is given by

$$
\begin{cases}
\begin{array}{rcl}
 & & \begin{matrix} & & n_j+1-t \\ & & \downarrow \end{matrix} \\
1+t & \to & \begin{bmatrix} 0 & & & & \\ \vdots & \cdot & & & \\ 1 & & \cdot & & \\ & \ddots & & \cdot & \\ & & 1 & \cdots & 0 \end{bmatrix}
\end{array}, \quad S_j \in \mathbf{PRO}(i) & (4.11a) \\[2ex]
\begin{array}{rcl}
 & & \begin{matrix} & & 1+t \\ & & \downarrow \end{matrix} \\
n_j+1-t & \to & \begin{bmatrix} 0 & \cdots & 1 & & \\ & \cdot & & \ddots & \\ & & \cdot & & 1 \\ & & & \cdot & \vdots \\ & & & & 0 \end{bmatrix}
\end{array}, \quad S_j \notin \mathbf{PRO}(i) & (4.11b)
\end{cases}
$$

Multiplying $\boldsymbol{S}_{\eta_j^i}$ from (4.11) by $\operatorname{diag}(\boldsymbol{\omega}_j^i) = \operatorname{diag}(\mathbf{dep}_j^i)$ from Equation (4.9) yields the matrix in $\mathbb{R}^{(n_j+1)\times(n_j+1)}$ given by

$$
\begin{cases}
\begin{array}{rcl}
 & & \begin{matrix} & & n_j+1-t \\ & & \downarrow \end{matrix} \\
1+t & \to & \begin{bmatrix} 0 & & & & \\ \vdots & \cdot & & & \\ (\boldsymbol{\omega}_j^i)_1 & & \cdot & & \\ & \ddots & & \cdot & \\ & & (\boldsymbol{\omega}_j^i)_{n_j+1-t} & \cdots & 0 \end{bmatrix}
\end{array}, \quad S_j \in \mathbf{PRO}(i) & (4.12a) \\[2ex]
\begin{array}{rcl}
 & & \begin{matrix} & & 1+t \\ & & \downarrow \end{matrix} \\
n_j+1-t & \to & \begin{bmatrix} 0 & \cdots & (\boldsymbol{\omega}_j^i)_{1+t} & & \\ & \cdot & & \ddots & \\ & & \cdot & & (\boldsymbol{\omega}_j^i)_{n_j+1} \\ & & & \cdot & \vdots \\ & & & & 0 \end{bmatrix}
\end{array}, \quad S_j \notin \mathbf{PRO}(i) & (4.12b)
\end{cases}
$$

and we see from Equations (4.10) and (4.12) that

$$\boldsymbol{S}_{\eta_j^i} * \text{diag}(\boldsymbol{\omega}_j^i) = (\boldsymbol{E}_j^{(i)})^T.$$

Thus, we see that the off-diagonal elements of $\boldsymbol{A}$ are equivalent to the off-diagonal elements of $\boldsymbol{Q}^T$.

4.3.3 Tensor Train

The work presented thus far has discussed the comparison of the CME operators developed in the tensor and SAN representations; it was also noted that Kazeev et al. [4] importantly used tensor train decomposition to further compress the Kronecker products of Hegland et al. [3]. An open question that the work did not address is if the savings of the tensor train compressions of Kazeev et al. [4] applied to the SAN formulation of Wolf [9] would be more economical or less economical than to its application to Hegland et al. [3]. Below is a brief background on tensor train to lay the groundwork towards investigating that problem (note that the tensor train presented here is demonstrated for illustrative purposes; the decomposition incorporated into [4] is the *quantized tensor train* variant).

In the tensor train decomposition (presented in [6] and used in reference to the CME in [4, 2]) of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, one is able to compute the value of $\mathcal{A}_{i_1,i_2,\ldots,i_N}$ through a series of vector and matrix multiplications where the vectors and matrices to be multiplied come from a new set of N tensors. In these multiplications, the i_n^{th} slices of the n^{th} tensors are multiplied with each other to attain element $\mathcal{A}_{i_1,i_2,\ldots i_N}$. The sizes of the dimensions of the component tensors are chosen such that a scalar is attained [4]; for a more detailed description of the functionality of tensor train, refer to [6].

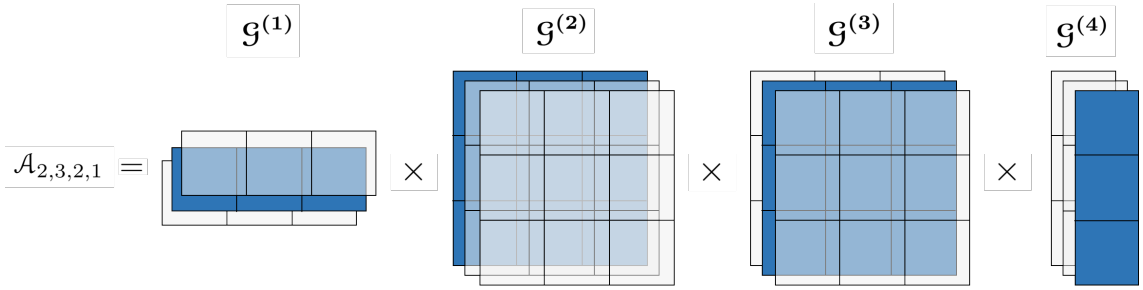


Figure 8: Visual representation of Tensor Train Decomposition.

In Figure 8, a 4-dimensional tensor $\boldsymbol{\mathcal{A}}$ has been decomposed into four 3-dimensional tensors, $\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)}$, and $\mathcal{G}^{(4)}$. The value at $\mathcal{A}_{2,3,2,1}$ is calculated by multiplying the row-vector from the 2^{nd} frontal slice of $\mathcal{G}^{(1)}$, the matrix from the 3^{rd} frontal slice of $\mathcal{G}^{(2)}$, the matrix from the 2^{nd} frontal slice of $\mathcal{G}^{(3)}$, and the column vector from the 1^{st} frontal slice of $\mathcal{G}^{(4)}$.

Accessing elements of the original tensor is computationally intensive for a tensor with many dimensions. However, tensor train is less storage intensive than storing the original tensor, and hence is very promising in CME applications, where tensor representations can become very large. Given that Kazeev et al.'s work added to the efficiency of Hegland et al.'s method via the introduction of tensor train decompositions, the proven similarity presents motivation to explore the addition of tensor train decompositions to our modified version of Wolf's algorithm.

5 Conclusion

Building on the more commonly known foundation of matrices, we are able to extend to multi-linear algebra using higher-order tensors. As our presentation showed, doing so is beneficial for the representation of big data sets of high dimensions, but comes with its own set of challenges, particularly with application to the chemical master equation. Our 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. We wrapped our comparison by raising the open question of how compression methods would have fared if the SAN approach was instead used in subsequent works such as Kazeev et al.

Having explored a potentially close relationship between stochastic automata networks and higher-order tensors, this work motivates exploring higher-order tensors in other stochastic automata problems and continuous time Markov processes outside of chemical reaction systems. This is very important as we see these types of problems appearing in nearly all fields, from physics to neuroscience that we seek to investigate by incorporating tensors and tensor decompositions. Furthermore, there are other problems within the application focus of chemical reaction systems. One such problem that arises in the modeling of biological systems is not just finding the probabilities of being in certain states after a time interval, but also finding the marginal probabilities of having certain quantities of specific molecules regardless of what other molecules are present (so, regardless of the state as a whole). In general, tensor decomposition shows great potential for large-data representation and modeling.

References

[1] A. Aho, M. Lam, R. Sethi, and J. Ullman, *Compilers: Principles, Techniques, and Tools*, Pearson Addison–Wesley, 2nd ed., 2007.

[2] S. Dolgov and B. Khoromskij, *Simultaneous state-time approximation of the chemical master equation using tensor product formats*, Numerical Linear Algebra with Applications, 22 (2015), https://doi.org/10.1002/nla.1942.

[3] M. Hegland and J. Garcke, *On the numerical solution of the chemical master equation with sums of rank one tensors*, ANZIAM Journal, 52 (2011), https://doi.org/10.21914/anziamj.v52i0.3895.

[4] V. Kazeev, M. Khammash, M. Nip, and C. Schwab, *Direct solution of the chemical master equation using quantized tensor train*, PLOS Computational Biology, 10 (2014).

[5] T. Kolda and B. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.

[6] I. Oseledets, *Tensor-train decomposition*, SIAM J. Scientific Computing, 33 (2011), pp. 2295–2317, https://doi.org/10.1137/090752286.

[7] H. Qian and L. Bishop, *The chemical master equation approach to nonequilibrium steady-state of open biochemical systems: linear single-molecule enzyme kinetics and nonlinear biochemical reaction networks.*, International Journal of Molecular Sciences, 11 (2010), pp. 3472—-3500, https://doi.org/10.3390/ijms11093472.

[8] H. D. Vo and R. B. Sidje, *An adaptive solution to the chemical master equation using tensors*, The Journal of Chemical Physics, 147 (2017), p. 044102, https://doi.org/10.1063/1.4994917.

[9] V. Wolf, *Modelling of biochemical reactions by stochastic automata networks*, Electron. Notes Theor. Comput. Sci., 171 (2007), pp. 197—208, https://doi.org/10.1016/j.entcs.2007.05.017.