

UM-DAE CENTRE FOR EXCELLENCE IN BASIC
SCIENCES

SCHOOL OF MATHEMATICAL SCIENCE

MAX PLANCK INSTITUTE OF MOLECULAR CELL
BIOLOGY AND GENETICS, DRESDEN

Learning Chemical Reaction Networks (CRNs)

Supervisor

Prof. Ivo F. SBALZARINI,
Chair of Scientific Computing
for Systems Biology,
TU Dresden & Max Planck
Institute of Molecular Cell
Biology and Genetics, Dresden

Author

Vishvas RANJAN,
UM DAE Centre for Excellence
in Basic Sciences, Mumbai

Dr. Abhishek BEHERA,
Max Planck Institute of
Molecular Cell Biology &
Genetics, Dresden



MAX PLANCK INSTITUTE
OF MOLECULAR CELL BIOLOGY
AND GENETICS



JUNE - DEC 2024
DRESDEN, GERMANY

CSBD, I.F. Sbalzarini, Pfotenhauerstr. 108, 01307 Dresden, Germany



center for
systems biology
dresden

Prof. Dr. Ivo F. Sbalzarini
Chair of Scientific Computing
for Systems Biology

Dresden, 28 May 2025

Confirmation of Internship

This is to confirm that Mr. Vishvas Ranjan completed an internship in the research group of Prof. Dr. Ivo F. Sbalzarini in the period from June 12, 2024, until December 23, 2024. Our research group is located at the Center for Systems Biology Dresden, jointly affiliated with the Faculty of Computer Science of TUD | Dresden University of Technology and with the Max Planck Institute of Molecular Cell Biology and Genetics. During his internship here, Vishvas worked on his Masters Thesis titled "Learning Chemical Reaction Networks".

Prof. Dr. Ivo F. Sbalzarini



Abstract

In this thesis learning at the cellular scale is considered, for example in simpler life forms like single-celled organisms. At this scale learning happens chemically and such systems can be abstracted out as a "soup" or "bag" of interacting chemicals. The mathematical framework for such interactions are the Chemical Reaction Networks (CRNs).

In this project we aim to demonstrate (mathematically) that such chemical systems can learn by showing that rate constants of stochastic chemical reaction networks can be tuned to represent arbitrary probability distributions. We do this by parametrizing the stationary distribution of CRNs as a function of their reaction rates using Markov Chain Tree Theorem (MCTT) and attempting to learn these parameters using KL-divergence between a target distribution and our model.

We first verify MCTT form for stationary distribution against Gillespie Algorithm and achieve $1/\sqrt{n}$ convergence. We attempt to learn the reaction rates using various methods such as brute force combinatorial search, automatic differentiation and gradient descent. While our results demonstrate that learning small chemical systems are possible, we face scalability challenges. We posit that additional structures on chemical reaction networks are necessary to make learning computationally tractable. We discuss some additional structures that may be imposed on the chemical reaction networks to make learning tractable.

Acknowledgement

I would like to express my deepest gratitude to my project supervisor, **Prof. Ivo F. Sbalzarini**, and co-supervisor, **Dr. Abhishek Behera**, for providing me with the opportunity to work on this fascinating topic. Their support has been instrumental in sharpening my mathematical modeling and programming skills to an exceptional level. I am grateful to Prof. Ivo for creating a comfortable and inspiring work environment.

I would like to thank my professors, **Dr Ameeya Bhagawat**, **Prof SG Dani**, **Prof MS Raghunathan** and **Dr Swagata Sarkar**, for instilling in me a strong foundation in mathematics.

I owe my deepest thanks to my family: my sister, **Anshika Ranjan**; my father, **Mr Ashish Ranjan**; and my mother, **Mrs Anjana Srivastava**, for their unwavering support, love, and guidance. Their dedication has made me capable of pursuing my dreams and carving my own path. Their hard work allowed me to experience the dynamic environment of **Sbalzarini Lab, MPI CBG**, where I was exposed to a rich blend of mathematics and computer science, an environment I could only imagine at the start of my higher studies.

Many many thanks to **Dr Abhishek**, who has been both a mentor and a **big brother**, guiding me on my academic journey. I also thank **Dr Nandu Gopan** and **Lucie Weigelt** for their invaluable assistance in processing my documents.

A special thanks to **Robinaresh Rathore (Robi)** for making my stay in Germany memorable by introducing me to the German culture and providing me with invaluable support outside academics. I would also like to express my appreciation to my cousin brother **Ankit Srivastava** and his wife for offering me a comfortable place in Berlin, ensuring I never missed home.

I would like to extend a special thanks to the "**School of Embedded Composite Artificial Intelligence**" and the "**DAAD's Konrad Zuse School of Excellence**" for sponsoring my stay in Dresden. My heartfelt thanks go to **Syam Prakash Manohar** and **Abhinav Singh** for their academic support during my time there.

I am thankful to my colleagues at MPI-CBG, including **Thomas, Roua, Dominik, Juan, Sina, Kiran, Justina**, and many others, for fostering a healthy and collaborative working environment. Many thanks to my neighbors, **Varun** and **Naman**, for making my living space a happy one.

I would like to thank my friends, **Saket, Sandip**, and **Varun**, for staying in touch and supporting me throughout. A special mention to **Nora** for making my last few months memorable during the stressful times of finalizing results.

Finally,

To those whom I may have missed, know that you hold a special place in my heart.

Contents

1	Background and Motivation	6
1.1	The Necessity of Modeling Learning at the Cellular Level	6
1.2	Learning and Sampling: Two Directions of Exploration	6
1.2.1	Inference: Can We Decode Hidden Dynamics?	6
1.2.2	Sampling: Can CRNs Generate Complex Patterns?	7
1.3	Critique of Turing Patterns	7
1.4	Encoding Diffusion in Purely Temporal Terms	7
1.5	Challenges of Spatial Representation in CRNs	7
1.6	Inspiration and Broader Implications	8
2	Markov Chain Tree Theorem	9
2.1	Graph Theory Basics	9
2.2	Markov Chain	10
2.2.1	Definition	10
2.2.2	Markov Matrix and Properties	11
2.2.3	Example: 4-Node Markov Chain	11
2.2.4	Distribution Vector	11
2.2.5	Stationary Distribution	12
2.2.6	Laplacian of a Markov Matrix	12
2.3	Calculating stationary distribution for a markov chain	12
2.3.1	Finding the Stationary Distribution	13
2.4	Theorem 1	14
2.5	Markov Chain Tree Theorem	15
2.6	Stationary Distribution as the Adjugate of Laplcian	15
2.7	Adjugate Form of MCTT	15
3	Chemical Reaction Networks (CRNs)	16
3.1	Definition	16
3.1.1	Stochastic Chemical Reaction Networks	16
3.1.2	Propensity Functions	16
3.1.3	State Transitions	17
3.1.4	Reachability Class	17
3.2	Continuous-Time Markov Chains (CTMCs)	17
3.2.1	Chemical Master Equation	18
3.2.2	Stationary Distribution of a CRN	18
3.3	CRNs induce CTMC on Discrete Lattice	18
3.3.1	Reachability Class on Discrete Lattice	19
3.4	Conjecture: Boundedness of Reachability Classes	19
3.5	Connection between CRNs and MCTT	20
3.6	Examples and Their Analysis Using MCTT	21

3.6.1	Simple Reversible Reaction Network	21
3.6.2	Simple Reversible Reaction with Three Species	24
3.7	Remark	25
4	Verification of MCTT	26
4.1	Verification of MCTT by Product-form Stationary Distribution	26
4.2	Verification of MCTT by Gillespie Algorithm	27
4.2.1	Computational Approach	27
4.2.2	Example 1: Simple Reversible CRN $A \xrightleftharpoons[k_1]{k} B$	28
4.2.3	Example 2: Enzyme-Catalytic Reaction	29
4.2.4	Remark	30
5	Learning Using CRNs	31
5.1	Learning Rule using MCTT	31
5.2	Learning using <i>argmin</i> method	32
5.2.1	Using <i>argmin</i> on $A \xrightleftharpoons[k_1]{k} B$	32
5.2.2	Using <i>argmin</i> on Enzyme-Catalysis: $E + S \xrightleftharpoons[k_1]{k} ES$	34
5.2.3	Limitations	35
5.3	Learning using <i>gradient-descent</i> with Automatic Differentiation	35
5.3.1	Automatic Differentiation (AD)	36
5.3.2	AD in $A \xrightleftharpoons[k_1]{k} B$	36
5.3.3	AD in Enzyme-Catalysis: $E + S \xrightleftharpoons[k_1]{k} ES$	37
5.4	Learning from Energy-Based models	38
5.4.1	Energy-Based Models	38
5.4.2	Three Node Energy-Based Model	39
5.4.3	Significance of Learning CRNs from EBMs	41
5.4.4	Considerations and Possible Challenges	41
5.5	Learning CRN Parameters from Data	42
5.5.1	Construction of CRN	42
5.5.2	Stationary Distribution and Marginalization	42
5.5.3	Defining Loss-function of Data	43
5.5.4	Computing Gradient	43
5.5.5	Sampling-based computation of Gradient	45
5.5.6	Updating Parameter and Convergence	46
5.6	Example using Learning Rule	46
5.7	Key Challenges with CRN-based Learning	48
6	Structures in CRN and Future Works	49
6.1	Challenges with Determinant in the Learning Rule	49

6.2	Sparse Representations	49
6.2.1	Structural Patterns in Sparse Matrices	49
6.3	Redefining Stationary Distributions Through Complex Balanced CRNs	50
6.3.1	Complex Balanced Condition	50
6.3.2	Stationary Distribution of a Complex Balanced CRN	51
6.4	Structural Approaches to Enforce Complex Balanced Condition	51
6.4.1	Structure on Parameters	51
6.4.2	Specific Structure in CRN: 0-deficiency CRN	51
6.5	TCBM Structure and Deficiency	52
6.5.1	Advantages of Zero-Deficiency CRNs	52

स्वयं एवं अपने परिवार के प्रति समर्पित एक प्रयास – सादर वंदन!

Ein Hoch auf mich und meine Familie – Prost!

Jun-Dec, 2024

LEARNING CHEMICAL REACTION NETWORKS

Master's Thesis

Vishvas Ranjan

Chapter 1: Background and Motivation

How do living systems learn and adapt? How can abstract mathematical models help us unravel the complex interplay of molecules that define life? These questions form the cornerstone of this thesis, which explores the potential of *chemical reaction networks* (CRNs) as a framework to model and understand the dynamics of biological and chemical systems. CRNs, a collection of interacting chemical species governed by deterministic or stochastic rules, offer a bridge between the microscopic realm of molecular interactions and the macroscopic behaviors that shape the natural world.

This abstraction is more than theoretical; it is a gateway to addressing profound challenges in biology, chemistry, and computation. From the processes governing cellular life to the design of artificial intelligence systems inspired by nature, the ability to model and learn through CRNs opens new avenues for exploration. This chapter lays the foundation for the questions and challenges that drive this work:

- Can chemical systems truly "learn" in a mathematical sense?
- How can CRNs capture spatial dynamics, such as the diffusion of molecules?
- What are the limits of CRNs in representing complex systems, and how might we overcome them?

1.1 The Necessity of Modeling Learning at the Cellular Level

Nature is an unparalleled innovator. Biological systems, such as cells, inherently learn and adapt through chemical interactions—be it in signal transduction pathways, genetic regulation, or metabolic reactions. These processes occur chemically, but their essence can be abstracted mathematically. Imagine a "bag" of interacting chemicals, where the network's behavior depends on the rate constants of the reactions. This abstraction allows CRNs to approximate arbitrary probability distributions, enabling them to "learn" patterns in a mathematically precise sense.

1.2 Learning and Sampling: Two Directions of Exploration

The promise of CRNs extends beyond static modeling. This thesis explores two main directions: *learning through inference* and *pattern generation through sampling*.

1.2.1 Inference: Can We Decode Hidden Dynamics?

In many biological systems, only a subset of species and interactions are observable. For example, fluorescence microscopy reveals the behavior of tagged proteins but leaves the untagged ones invisible. How can we uncover the hidden dynamics of such systems? This challenge drives the field of *inference*, where the goal is to estimate the parameters of a CRN from partial data.

Using tools like the Markov Chain Tree Theorem and gradient-based optimization, this thesis demonstrates how reaction rates can be inferred, enabling the reconstruction of unobserved processes. By addressing questions of hidden variables, this work sheds light on the unseen aspects of biological systems and provides a framework for studying incomplete datasets.

1.2.2 Sampling: Can CRNs Generate Complex Patterns?

While inference decodes, *sampling generates*. Inspired by advances in generative artificial intelligence, CRNs can be tuned to produce specific patterns, such as Turing patterns [1] observed in natural systems. This thesis explores the potential of CRNs to approximate target probability distributions by optimizing reaction rates through gradient descent and automatic differentiation.

For example, imagine creating a CRN capable of generating intricate biological patterns or simulating molecular distributions. Although current methods do not match the scale of AI models generating MNIST datasets, they demonstrate the feasibility of using CRNs for probabilistic sampling. However, challenges in scalability and precision remain. How can we scale these methods to more complex systems? And can we harness the computational power of CRNs to rival traditional AI techniques?

1.3 Critique of Turing Patterns

Turing patterns, classically understood, emerge from spatio-temporal dynamics where diffusion plays a pivotal role [2]. The interplay of reaction and diffusion leads to spatially heterogeneous patterns, famously used to model phenomena such as animal coat patterns or Zebrafish. However, a thought-provoking question arises:

Can we generate Turing patterns without explicitly incorporating spatial dynamics?

Chemical Reaction Networks serve as a universal language capable of expressing diverse phenomena. While their formulations may lack elegance or simplicity, they possess the power to construct circuits, simulate neural networks, and model processes like diffusion. Intriguingly, CRNs can also simulate cellular automata, underscoring their versatility. Among these varied capabilities, diffusion holds a special significance in the context of Turing patterns.

1.4 Encoding Diffusion in Purely Temporal Terms

Diffusion is crucial to the classical generation of Turing patterns. Yet, by leveraging the flexibility of CRNs, it is possible to encode the spatial aspect of diffusion entirely within temporal dynamics. This can be achieved by translating spatial diffusion into a larger set of chemical species, representing discrete locations or concentration gradients. The CRN then governs the interactions among these species, mimicking the effects of diffusion without relying on explicit spatial terms.

This abstraction demonstrates the power of CRNs to not only emulate diffusion but also extend the classical framework of Turing patterns. By operating solely in the temporal domain, CRNs can reproduce the characteristic patterns traditionally linked to spatial dynamics.

The ability to generate Turing patterns without spatial dynamics showcases the versatility of CRNs as a modeling framework. By encoding spatial effects temporally, CRNs can simulate complex biological and chemical systems where spatial representation is impractical. This aligns with our work on demonstrating how CRNs bridge classical spatial models and purely temporal frameworks, enabling innovative approaches to pattern formation and beyond.

1.5 Challenges of Spatial Representation in CRNs

Consider diffusion processes, where molecules move through space due to concentration gradients. These dynamics, often described by partial differential equations (PDEs), can be discretized

into finite elements, allowing CRNs to model the movement of molecules. In this framework, diffusion becomes a series of chemical reactions: a molecule "dies" at one location and is "born" at another. Such a translation, while computationally demanding, underscores the versatility of CRNs in representing spatial phenomena.

Yet, how can we reconcile the infinite nature of continuous space with the finite constraints of computation? Even within a bounded region, representing every point as a distinct chemical species would lead to an intractable system. To tackle this, we know there are discretization schemes, breaking space into manageable chunks. Each chunk corresponds to a chemical species, and diffusion is modeled as interactions within a finite radius of neighbors.

However, this simplification introduces its own challenges. Scaling these models to larger systems without losing fidelity is computationally expensive. Consider the example of a DNA strand diffusing through a 2D grid. At each point, the state of the DNA encodes a value—its "ON" or "OFF" status—and as it diffuses, this state propagates to neighboring locations. Expressing such behavior as a CRN translates diffusion dynamics into the language of chemical reactions, making it accessible for simulation and analysis. But how far can we push this approach? And what trade-offs must we make between accuracy and computational feasibility?

1.6 Inspiration and Broader Implications

This thesis takes a scientific yet exploratory approach, blending rigorous mathematical modeling with conceptual innovation. By addressing questions of *inference* and *sampling*, it contributes to a growing body of knowledge at the intersection of biology, chemistry, and computation.

- How can we refine CRNs to better approximate real-world systems?
- What new insights can we gain about biological learning by studying chemical systems?
- Can CRNs serve as the foundation for a new kind of computational framework, one inspired by the principles of life itself?

These questions form the heart of this work. By addressing the challenges of spatial representation, inference, and sampling, this thesis lays the groundwork for advancing our understanding of learning and computation in chemical systems. From decoding hidden dynamics to generating patterns, the insights gained here push the boundaries of what is possible with CRNs, opening doors to exciting future research.

Chapter 2: Markov Chain Tree Theorem

2.1 Graph Theory Basics

Weighted Directed Graph

A **weighted directed graph** G is defined as follows:

- A set of vertices $V(G)$.
- A set of directed edges $E(G)$, where each edge is an ordered pair (u, v) with $u, v \in V(G)$.
- A weight function $w : E(G) \rightarrow \mathbb{R}$ that assigns a real number to each edge.

Weight of a Graph

The **weight of a graph** is the sum of the weights of all its edges:

$$\text{Weight}(G) = \sum_{e \in E(G)} w(e)$$

Equivalence Relation

We define an equivalence relation \sim on the set of vertices $V(G)$ as follows:

$$u \sim v \iff \text{there is a walk from } u \text{ to } v \text{ and } v \text{ to } u$$

Strongly Connected Components

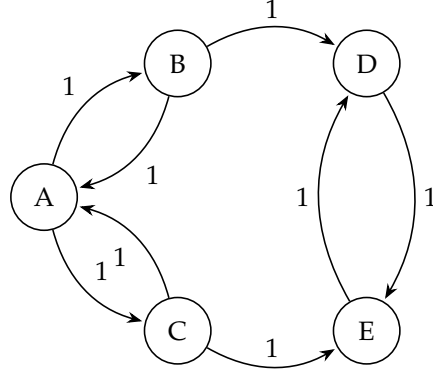
The equivalence relation \sim partitions $V(G)$ into equivalence classes called **strongly connected components**. Each strongly connected component (SCC) is a maximal subgraph where every pair of vertices u and v are reachable from each other.

Closed Class

A **closed class** is a subset of vertices such that no vertex in the subset has an outgoing edge to a vertex outside the subset.

Strongly Connected Component as a Closed Class

A strongly connected component is a closed class if and only if it has no outgoing edge to any vertex outside the component.



In the above diagram, the vertices $\{A, B, C\}$ form a strongly connected component since there are walks between any pair of these vertices in both directions. If this component has no outgoing edges to other vertices, it is also a closed class.

Lemma 2.1. *Every graph contains at least one closed class.*

Proof. The proof of this lemma can be found in [3]. □

Unichain Graph

A graph is called unichain if it has exactly one closed class.

Directed tree rooted at vertex

A graph $G = (V, E)$ is called a *directed tree rooted at* $v \in V$ if and only if G contains a unique walk from each vertex in V to v .

Spanning Tree

A *spanning tree* is a subset of the edges of G that connects all the vertices together without any cycles. Each connected graph has at least one spanning tree.

2.2 Markov Chain

A **Markov chain** is a stochastic process that undergoes transitions from one state to another on a state space. It is characterized by the property that the next state depends only on the current state and not on the sequence of states that preceded it. This property is called the **Markov property**.

2.2.1 Definition

A Markov chain can be described by:

- A finite set of states $S = \{s_1, s_2, \dots, s_n\}$.
- A transition matrix P where each entry P_{ij} represents the probability of transitioning from state s_i to state s_j , and the sum of the entries in each row is 1, i.e., $\sum_j P_{ij} = 1$.

2.2.2 Markov Matrix and Properties

A **Markov matrix** (also known as a transition matrix) M is a square matrix used to describe the transitions of a Markov chain. Each element M_{ij} of the matrix represents the probability of transitioning from state i to state j .

Properties:

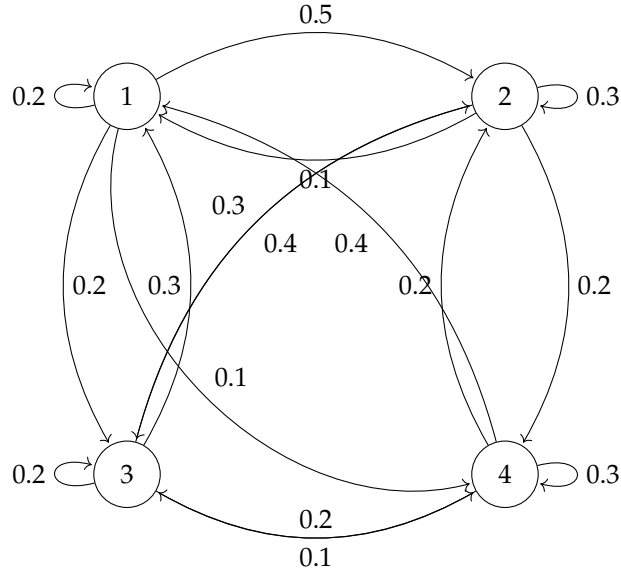
- **Non-negativity:** All entries of M are non-negative, i.e., $M_{ij} \geq 0$ for all i, j .
- **Row Sum to One:** The sum of the entries in each row of M is equal to one, i.e., $\sum_j M_{ij} = 1 \forall i$.

2.2.3 Example: 4-Node Markov Chain

Consider a Markov chain with 4 states: $S = \{1, 2, 3, 4\}$. The transition matrix P is given by:

$$P = \begin{pmatrix} 0.2 & 0.5 & 0.2 & 0.1 \\ 0.1 & 0.3 & 0.4 & 0.2 \\ 0.3 & 0.3 & 0.2 & 0.2 \\ 0.4 & 0.2 & 0.1 & 0.3 \end{pmatrix}$$

The graph of this markov chain, $G(P)$ will be:



2.2.4 Distribution Vector

A **distribution vector** $\mathbf{p} \in \mathbb{R}^n$ is a row vector that represents the probabilities of being in each state of the Markov chain. The properties of the distribution vector are:

- **Non-negativity:** $\mathbf{p}_i \geq 0$ for all i .
- **Sum to One:** $\sum_i \mathbf{p}_i = 1$.

The relation of the distribution vector at the next step using the Markov matrix is given by:

$$\mathbf{p}^{(t+1)} = \mathbf{p}^{(t)} M$$

2.2.5 Stationary Distribution

A distribution vector π is called a **stationary** or **stable distribution** if it satisfies the equation:

$$\pi M = \pi$$

and the properties of the distribution vector:

$$\sum_i \pi_i = 1$$

The stationary distribution represents the long-term behavior of the Markov chain, where the distribution of states remains constant over time.

2.2.6 Laplacian of a Markov Matrix

Given a Markov matrix P , the Laplacian matrix Λ is defined as:

$$\Lambda = P - I$$

where I is the identity matrix.

- Λ is a singular matrix.
- The sum of each row of Λ is zero.
- The kernel of Λ (i.e., the set of vectors \mathbf{x} such that $\Lambda \mathbf{x} = 0$) includes the stable distribution vector of P .

2.3 Calculating stationary distribution for a markov chain

For a Markov matrix P , a stationary distribution vector π satisfies:

$$\pi P = \pi$$

This implies:

$$\pi(P - I) = \pi(0) = 0$$

Hence, π is in the kernel of Λ . Therefore, the stationary distribution vector is in the kernel of the Laplacian of P .

- The number of strongly connected components in the directed graph represented by P is equal to the dimension of the kernel of $\Lambda = P - I$.
- This is because each strongly connected component contributes a linearly independent vector to the kernel, corresponding to the stationary distribution of that component.

Consider the Markov matrix P and graph $G(P)$ from 2.2.3. The Laplacian of this matrix P is given by:

$$\Lambda = P - I = \begin{pmatrix} -0.8 & 0.5 & 0.2 & 0.1 \\ 0.1 & -0.7 & 0.4 & 0.2 \\ 0.3 & 0.3 & -0.8 & 0.2 \\ 0.4 & 0.2 & 0.1 & -0.7 \end{pmatrix}$$

2.3.1 Finding the Stationary Distribution

- **Method 1: Solving System of Equations**

To find the stationary distribution vector π , we solve $\Lambda\pi = 0$ subject to $\sum_i \pi_i = 1$ and $\pi_i \geq 0$.

$$\Lambda\pi = \begin{pmatrix} -0.8 & 0.5 & 0.2 & 0.1 \\ 0.1 & -0.7 & 0.4 & 0.2 \\ 0.3 & 0.3 & -0.8 & 0.2 \\ 0.4 & 0.2 & 0.1 & -0.7 \end{pmatrix} \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Solving this system while keeping $\sum_i \pi_i = 1$, we find the stationary distribution vector π is:

$$\pi = \left(\frac{40}{173} \quad \frac{113}{346} \quad \frac{85}{346} \quad \frac{34}{173} \right)$$

- **Method 2: Initial State to Stationary Distribution**

We can also find the stationary distribution using way of convergence. Starting with an initial state vector \mathbf{x}_0 , the state vector at step n is given by $\mathbf{x}_n = \mathbf{x}_0 P^n$. As $n \rightarrow \infty$, \mathbf{x}_n converges to the stationary distribution π . For example, if the initial state vector is $\mathbf{x}_0 = (1, 0, 0, 0)$, we compute \mathbf{x}_n iteratively until convergence to the stationary distribution.

- **Method 3: Using Adjugate Matrix**

The space of stationary or stable vectors of a unichain Markov matrix P has dimension 1. We can describe the space of stationary vectors using the adjugate matrix of P . We will discuss it in the rest of this section.

Lemma 2.2. *If P is a Markov matrix whose graph $G(P)$ has k closed classes, then $\dim(\ker(P - I)) = k$. That is, the space of stable vectors of P has dimension k .*

The lemma is mentioned in [3] as theorem 3.1.

Minor and Cofactor

Given a matrix A , the minor $M_{ij}(A)$ is the matrix formed from A by deleting row i and column j . The cofactor $C_{ij}(A)$ is defined as:

$$C_{ij}(A) = (-1)^{i+j} \det(M_{ij}(A))$$

Adjugate Matrix

The **adjugate matrix** $\text{adj}(P)$ of a square matrix P of order n is defined by:

$$(\text{adj}(M))_{ij} = C_{ji}(M)$$

where $C_{ji}(M)$ is the cofactor of M at position (j, i) .

Lemma 2.3. For any $n \times n$ matrix P , we have:

$$P \cdot \text{adj}(P) = \text{adj}(P) \cdot P = \det(P) \cdot I$$

where $\det(P)$ is the determinant of P , and I is the $n \times n$ identity matrix. Therefore, $\text{adj}(P)$ is the adjugate matrix of P , and $\det(P)$ is the determinant of P .

2.4 Theorem 1

Given a unichain Markov matrix P with Laplacian $\Lambda = P - I$, the vector \mathbf{v}_P whose entries are the diagonal entries of the adjugate of Λ , $(\mathbf{v}_P)_i = (\text{adj}(\Lambda))_{ii}$, is a stable or stationary vector of P . That is, $P\mathbf{v}_P = \mathbf{v}_P$.

Proof. This theorem is mentioned in [3] with a proof as Theorem 3.4.

Since $\Lambda \text{adj}(\Lambda) = \det(\Lambda)$. By Lemma 2.2, the off-diagonal entries of $\Lambda \text{adj}(\Lambda)$ are 0 and the diagonal entries are $\det(\Lambda)$. Since P is a unichain Markov matrix, then by Theorem 1 in 2.4, the Laplacian matrix Λ has $\dim(\ker(\Lambda)) = 1$. As a result, Λ has non-trivial kernel and hence $|\Lambda| = 0$.

Since the determinant of Λ is zero, the matrix product $\Lambda \text{adj}(\Lambda)$ is the **zero matrix**. This means that every column of $\text{adj}(\Lambda)$ lies in $\ker(\Lambda)$, i.e., every column of the adjugate of Λ is a solution to $\Lambda v = 0$.

The columns of Λ , corresponding to the transition matrix P , sum to 0 because each row of P represents a probability distribution that sums to 1, and subtracting the identity matrix gives rows that sum to 0. This means that the rows of the transpose of Λ , denoted Λ^T , also sum to 0.

Let J be the vector of all 1s. Then, since the rows of Λ^T sum to 0, it follows that:

$$\Lambda^T J = 0,$$

meaning that $J \in \ker(\Lambda^T)$, i.e., J lies in the kernel of Λ^T .

Since Λ is a square matrix and $\dim(\ker(\Lambda)) = 1$, by symmetry, $\dim(\ker(\Lambda^T)) = 1$ as well. This means that every vector in $\ker(\Lambda^T)$ is a multiple of J .

Again, considering the product $\text{adj}(\Lambda)\Lambda$, this is the **zero matrix**. Thus, $(\text{adj}(\Lambda)\Lambda)^T = \Lambda^T \text{adj}(\Lambda)^T = 0$. This implies that every column of $\text{adj}(\Lambda)^T$ (i.e., every row of $\text{adj}(\Lambda)$) lies in $\ker(\Lambda^T)$, and therefore, each row of $\text{adj}(\Lambda)$ must be a multiple of J .

Since each row of $\text{adj}(\Lambda)$ is a multiple of J , all column vectors of $\text{adj}(\Lambda)$ must be equal. These vectors are all equal to the vector v_P , whose entries are the **diagonal entries** of $\text{adj}(\Lambda)$.

Since $v_P \in \ker(\Lambda)$, it satisfies $\Lambda v_P = 0$. Recalling that $\Lambda = P - I$, we conclude that:

$$(P - I)v_P = 0 \implies Pv_P = v_P.$$

Thus, v_P is a **stable** or stationary vector of P , meaning that it remains unchanged under the application of the Markov matrix P . \square

2.5 Markov Chain Tree Theorem

For each state i , the stationary probability π_i is proportional to the total weight of all spanning trees of the Markov chain graph rooted at i . Specifically:

$$\pi_i = \frac{T_i}{\sum_{j=1}^n T_j},$$

where T_i is the sum of the weights of all spanning trees rooted at state i .

Proof:

A detailed elementary proof of this theorem can be found in [3].

2.6 Stationary Distribution as the Adjugate of Laplacian

In the above theorem, we see how the **Markov Chain Tree Theorem (MCTT)** provides a graph-theoretic method for computing the stationary distribution vector of a Markov chain. Specifically, it utilizes the structure of the graph $G(P)$ associated with the transition matrix P . In the context of a unichain Markov graph, **Theorem 1** establishes a direct connection between the stationary distribution vector π , representing the steady-state probabilities of the Markov matrix P , and the *adjugate* of the *Laplacian matrix* Λ , where $\Lambda = P - I$. So, we can have the adjugate form of the previous theorem.

2.7 Adjugate Form of MCTT

The **Adjugate form of the Markov Chain Tree Theorem** states that the stationary distribution vector π is proportional to the diagonal entries of the adjugate matrix $\text{adj}(\Lambda)$. Mathematically, this relationship can be expressed as:

$$\pi_i \propto (\text{adj}(\Lambda))_{i,i}, \quad \forall i \tag{1}$$

where $(\text{adj}(\Lambda))_{i,i}$ denotes the i -th diagonal entry of the adjugate matrix.

Remark:

This result simplifies the computation of the stationary distribution vector by leveraging matrix operations on Λ , avoiding the explicit enumeration of spanning trees required in the original form of MCTT. The relationship highlights that each component π_i of the stationary vector π is determined by the corresponding diagonal entry of the adjugate matrix. Consequently, the stable distribution vector π , which characterizes the long-term behavior of the Markov process, can be computed directly from the graph-theoretic properties of the transition matrix encoded in its Laplacian Λ . This adjugate-based approach retains the structural insights of MCTT while being computationally efficient.

Chapter 3: Chemical Reaction Networks (CRNs)

Chemical Reaction Networks (CRNs) are mathematical frameworks that model the interactions and dynamics of chemical species through a set of reactions.

3.1 Definition

We define a **Chemical Reaction Network (CRN)** as a triplet $C = (S, R, \kappa)$, where:

- S is a non-empty finite set of species,
- R is a set of reactions, and
- κ is the associated set of reaction rate constants for each reaction in R .

Let $S = \{S_1, S_2, \dots, S_M\}$ be a finite set of M species. A **reaction** $r \in R$ is represented as a formal chemical equation:

$$\sum_{i=1}^M \mu_i^r S_i \rightarrow \sum_{i=1}^M \nu_i^r S_i$$

or, in abbreviated form, $r = \mu^r \rightarrow \nu^r$, where:

- $\mu^r, \nu^r \in \mathbb{N}^S$ are the **stoichiometric coefficient vectors** for the reactants and products, respectively.
- Here, $\mathbb{N} = \mathbb{Z}_{\geq 0}$, the set of non-negative integers.
- Each reaction r has an associated **reaction rate constant** $k_r \in \mathbb{R}_{>0}$, a positive real number that influences the speed of the reaction given by:

$$k_r : R \rightarrow \mathbb{R}_{\geq 0}$$

3.1.1 Stochastic Chemical Reaction Networks

In the context of a **stochastic chemical reaction network (CRN)**, the state of the system is described by a vector $\mathbf{s} = (s_1, s_2, \dots, s_M)$, where $s_i \in \mathbb{N}$ represents the discrete count of species S_i in the network. These states evolve according to a **discrete-time stochastic process**. The transitions between states are governed by reactions, each with a corresponding *propensity function* [3.1.2](#) that quantifies the likelihood of a reaction occurring in a given state.

3.1.2 Propensity Functions

The probability that a given reaction occurs per unit time, called its **propensity**. The propensity of a reaction r at a state \mathbf{s} is denoted by $\rho_r(\mathbf{s})$. It is computed as:

$$\rho_r(\mathbf{s}) = k_r \prod_{i=1}^M \frac{s_i!}{(s_i - \mu_i^r)!}$$

where:

- k_r is the reaction rate constant for reaction r ,
- μ_i^r is the stoichiometric coefficient of species S_i in the reactants of reaction r ,
- $\rho_r(\mathbf{s}) = 0$ if $s_i < \mu_i^r$, meaning the reaction cannot proceed if the number of molecules of a reactant species is insufficient.

The term $\prod_{i=1}^M \frac{s_i!}{(s_i - \mu_i^r)!}$ ensures that the propensity depends on the combinatorial ways in which reactants can collide to trigger the reaction.

This form of the propensity function assumes mass-action kinetics, where reactions are proportional to the product of the reactant concentrations. However, this simplification does not account for more complex kinetics, such as **Hill functions** or **Michaelis-Menten mechanisms**. This limitation can be addressed in future extensions by incorporating more generalized kinetic forms.

3.1.3 State Transitions

Each time a reaction r occurs, the system transitions from state \mathbf{s} to a new state $\mathbf{s} + \Delta_r$, where:

$$\Delta_r = \nu_r - \mu_r$$

is the *reaction vector*. Here:

- ν_r is the stoichiometric vector of products for reaction r ,
- μ_r is the stoichiometric vector of reactants for reaction r .

Thus, the propensity of each reaction changes as the state of the system evolves.

3.1.4 Reachability Class

The *reachability class* Ω_s of a given initial state \mathbf{s}_0 is defined as the set of all states $\mathbf{s} \in \mathbb{N}^M$ accessible via a sequence of reactions in the reaction network \mathcal{R} . The dynamics of the CRN are constrained to this subset of the state space.

3.2 Continuous-Time Markov Chains (CTMCs)

Viewed from the perspective of state transitions, stochastic CRNs can be modeled as *continuous-time Markov chains (CTMCs)*, where the transition rates are determined by the propensity functions. The transition rate from state \mathbf{s} to state \mathbf{s}' is given by:

$$R_{\mathbf{s} \rightarrow \mathbf{s}'} = \sum_{r: \mathbf{s}' = \mathbf{s} + \Delta_r} \rho_r(\mathbf{s}),$$

where the summation is taken over all reactions r that result in the transition $\mathbf{s} \rightarrow \mathbf{s}'$.

The transition rates forms the backbone of Markov chains and is crucial for modeling the stochastic dynamics of CRNs. It enables the computation of transition matrix and hence stationary distributions, providing insights into long-term species concentrations.

3.2.1 Chemical Master Equation

The evolution of the probability distribution $P(\mathbf{s}, t)$, which describes the probability of the system being in state \mathbf{s} at time t , is governed by the *chemical master equation*:

$$\frac{dP(\mathbf{s}, t)}{dt} = \sum_{\mathbf{s}' \neq \mathbf{s}} R_{\mathbf{s}' \rightarrow \mathbf{s}} P(\mathbf{s}', t) - R_{\mathbf{s} \rightarrow \mathbf{s}'} P(\mathbf{s}, t),$$

or equivalently:

$$\frac{dP(\mathbf{s}, t)}{dt} = \sum_{r \in \mathcal{R}} [P(\mathbf{s} - \Delta_r, t) \rho_r(\mathbf{s} - \Delta_r) - P(\mathbf{s}, t) \rho_r(\mathbf{s})].$$

This equation captures the net probability flux into and out of a given state \mathbf{s} .

3.2.2 Stationary Distribution of a CRN

A stationary distribution $\pi(\mathbf{s})$ satisfies:

$$\frac{dP(\mathbf{s}, t)}{dt} = 0, \quad \forall \mathbf{s}.$$

In general:

- For a given initial state s_0 , A stationary distribution $\pi(s) = P(s, \infty)$ may not exist or may not be unique.
- Uniqueness is guaranteed if the system has a single *reachability class* Ω_s , which is the maximal set of states accessible from the initial state \mathbf{s}_0 via the reaction network.

3.3 CRNs induce CTMC on Discrete Lattice

A Chemical Reaction Network (CRN) induces a *Continuous-Time Markov Chain (CTMC)* on the discrete lattice $\mathbb{Z}_{\geq 0}^s$ (representing the non-negative integer states of the species). The transition rates of this CTMC are given by $Q(k)$, where:

$$Q(k_r)(\mathbf{s} \rightarrow \mathbf{s}') = \sum_{\mathbf{y} \rightarrow \mathbf{y}' \in \mathcal{R}} k_r(\mathbf{y} \rightarrow \mathbf{y}') \binom{\mathbf{s}}{\mathbf{y}} \mathbf{y}!$$

subject to the following conditions:

- $\mathbf{s} \geq \mathbf{y}$: The current state \mathbf{s} must have at least as many molecules of each species as required by the reactant vector \mathbf{y} .
- $\mathbf{s}' - \mathbf{s} = \mathbf{y}' - \mathbf{y}$: The state transition is determined by the net stoichiometric change resulting from the reaction $\mathbf{y} \rightarrow \mathbf{y}'$.

Here:

- $k_r(\mathbf{y} \rightarrow \mathbf{y}')$ is the rate constant for the reaction $\mathbf{y} \rightarrow \mathbf{y}'$,
- $\binom{\mathbf{s}}{\mathbf{y}} \mathbf{y}!$ represents the multinomial coefficient, accounting for the combinatorial ways in which the reactants \mathbf{y} can be selected from the state \mathbf{s} .

3.3.1 Reachability Class on Discrete Lattice

Given an initial state \mathbf{s}_0 , the stationary distribution $\pi(\mathbf{s}) = P(\mathbf{s}, \infty)$ is unique if it exists. The set of states reachable from \mathbf{s}_0 is called the **reachability class**, denoted by $\Omega_{\mathbf{s}_0}$, and is defined as:

$$\Omega_{\mathbf{s}_0} = \mathcal{C}[\mathbf{s}_0] = \mathbf{s}_0 + \text{span}\{\mathbf{y}' - \mathbf{y} \mid \mathbf{y} \rightarrow \mathbf{y}' \in \mathcal{R}\} \cap \mathbb{Z}_{\geq 0}^s$$

Mathematically, it is constructed by adding the initial state \mathbf{s}_0 to the integer span of all reaction vectors $(\mathbf{y}' - \mathbf{y})$, intersected with the non-negative lattice $\mathbb{Z}_{\geq 0}^s$. The reachability class $\Omega_{\mathbf{s}_0} \subseteq \mathbb{N}^M$ is the maximal subset of the integer lattice accessible to the CRN.

3.4 Conjecture: Boundedness of Reachability Classes

A conjecture associated with the dynamics of CRNs states:

$$\text{No autocatalytic and No Birth reaction} \iff \mathcal{C}[\mathbf{s}_0] \text{ is bounded.}$$

This implies that:

- If there are no autocatalytic reactions (reactions in which a product species promotes its own production) and no birth reactions, the reachability class remains bounded.
- Conversely, the presence of autocatalytic reactions or birth may lead to unbounded growth in the reachability class.

Note:

We provide a formal proof; however, the term 'conjecture' reflects the simplicity of the system considered, leaving open the possibility of further exploration in more complex CRN configurations.

Proof:

1) No autocatalytic and No Birth reaction $\Rightarrow \mathcal{C}[\mathbf{s}_0]$ is bounded:

Consider a Lyapunov-like function $L(\mathbf{s})$ that measures the total "count" of species in the system at a state:

$$L(\mathbf{s}) = \sum_{i=1}^s w_i s_i,$$

where $w_i \geq 0$ are weights chosen based on the stoichiometry of the reactions.

For a reaction $\mathbf{y} \rightarrow \mathbf{y}'$ with stoichiometric vectors $\mathbf{y}, \mathbf{y}' \in \mathbb{Z}_{\geq 0}^s$, the state update is:

$$\mathbf{s} \rightarrow \mathbf{s}' = \mathbf{s} + (\mathbf{y}' - \mathbf{y}).$$

The change in $L(\mathbf{s})$ is:

$$\Delta L = L(\mathbf{s}') - L(\mathbf{s}) = \sum_{i=1}^s w_i (y'_i - y_i).$$

By given condition, if there are no autocatalytic reactions, the net production $\mathbf{y}' - \mathbf{y} \leq 0$ does not lead to unbounded growth. This implies that:

$$\Delta L \leq 0,$$

i.e., $L(\mathbf{s})$ is non-increasing or bounded. This means that there exists a global upper bound on $L(\mathbf{s})$. Since $L(\mathbf{s})$ is bounded, the set of reachable states $\mathcal{C}[\mathbf{s}_0]$ must also be bounded. Specifically, No species s_i can grow indefinitely because $\sum_{i=1}^S w_i s_i$ is bounded above.

Hence, If there are no autocatalytic reactions, $\mathcal{C}[\mathbf{s}_0]$ is bounded.

2) $\mathcal{C}[\mathbf{s}_0]$ is bounded \Rightarrow No autocatalytic and No Birth reaction:

We first assume $\mathcal{C}[\mathbf{s}_0]$ is bounded. This means there exists a finite bound $B > 0$ such that:

$$\|\mathbf{s}\| \leq B, \quad \forall \mathbf{s} \in \mathcal{C}[\mathbf{s}_0],$$

where $\|\mathbf{s}\|$ denotes the sum of the components of \mathbf{s} : $\|\mathbf{s}\| = \sum_{i=1}^M s_i$. Now, we assume there exists an autocatalytic reaction.

$$\sum_{i=1}^M \mu_i S_i \rightarrow \sum_{i=1}^M \nu_i S_i$$

This would imply, there exists a species S_j such that S_j is both a reactant and a product, with a **net gain** which results in the count of S_j in a state growing as $S_j \rightarrow n_1 S_j \rightarrow n_2 S_j \rightarrow n_3 S_j \rightarrow \dots$ with $n_1 < n_2 < n_3 \dots$ in increasing order.

If this reaction is in the system, starting from a nonzero initial state \mathbf{s}_0 , $\mathcal{C}[\mathbf{s}_0]$ becomes unbounded because S_j can grow without bound.

Contradiction! The assumption that $\mathcal{C}[\mathbf{s}_0]$ is bounded implies that no such unbounded growth is possible. Therefore, no autocatalytic reaction can exist.

Hence, If $\mathcal{C}[\mathbf{s}_0]$ is bounded, there are no autocatalytic reactions.

Thus, the conjecture is proved:

$$\text{No autocatalytic and No Birth reaction} \iff \mathcal{C}[\mathbf{s}_0] \text{ is bounded.}$$

3.5 Connection between CRNs and MCTT

We just saw that CRNs can be viewed as a subclass of CTMCs, where the underlying graph structure emerges naturally from the reaction network. Specifically, the connection lies in the fact that the state space of a CRN can be represented as a directed graph, where nodes correspond to states (molecular counts) and edges represent transitions governed by reaction propensities. The weight associated with each edge is directly related to the reaction rate constants, which define the underlying Markov chain. This connection allows the tools of Markov chains, particularly the MCTT, to be applied to CRNs. By interpreting CRNs as weighted directed graphs, it becomes possible to compute their stationary distributions using MCTT.

The stationary distribution of a CRN provides insights into its long-term behavior, such as the relative abundances of chemical species. Our first aim is to bridge the combinatorial structure of CRNs with their stochastic dynamics, focusing on the role of MCTT in deriving stationary distributions. Furthermore, the adjugate matrix form of MCTT offers computational advantages in analyzing CRNs, particularly when the stationary distribution needs to be expressed explicitly.

3.6 Examples and Their Analysis Using MCTT

To illustrate the connection between CRNs and MCTT, we now present two examples that demonstrate how the stationary distributions of CRNs can be computed using graph-theoretic tools. These examples serve as concrete realizations of the theoretical framework discussed and highlight the utility of MCTT in analyzing stochastic dynamics in chemical systems.

3.6.1 Simple Reversible Reaction Network

The first example considers a basic reversible reaction network, where two chemical species interconvert through a pair of forward and backward reactions. This network forms the simplest non-trivial CRN with a clear underlying graph structure. By applying MCTT, we will compute the stationary distribution of the system and compare the result to known equilibrium conditions derived from reaction rate constants.

Consider a simple reversible reaction:



with rate constants k for the forward reaction $A \rightarrow B$ and 1 for the reverse reaction $B \rightarrow A$. The system is constrained by the conservation law:

$$X_A(t) + X_B(t) = N$$

where $X_A(t)$ and $X_B(t)$ represent the number of molecules of species A and B , and N is the total number of molecules. This conservation law ensures that the state space is limited to $N + 1$ states, corresponding to possible values of $X_A = 0, 1, \dots, N$. Our state space will be discrete but on the line: $(0, N), (1, N - 1), \dots, (i, N - i), \dots, (N, 0)$ where i and $N - i$ are the concentrations of A and B respectively.

Transition Rates and Markov Chain:

The system can be modeled as a continuous-time Markov chain (CTMC), with:

- Forward transition rate from state i to $i + 1$ given by $k \times i$.
- Reverse transition rate from state $i + 1$ to i given by $1 \times (N - i)$.

The Laplacian of transition matrix P captures the rates of transitions between states and is constructed with:

$$\begin{aligned} P[i, i + 1] &= 1 \times (N - i), \\ P[i + 1, i] &= k \times i, \\ P[i, i] &= -(1 \times (N - i) + k \times i). \end{aligned}$$

Stationary Distribution π :

It can be computed by using MCTT since the state space is bounded as no autocatalytic reaction is involved here. This stationary distribution provides insight into the long-term proportions of A and B in the system.

Computational Approach:

We use the `Catalyst.jl` package in Julia to define the reactions and compute the stationary distribution. By varying the parameter k (the ratio of forward to backward rate constants) and N (the total number of molecules), we can observe the system's behavior under different conditions. We also use `Symbolics.jl` and `LinearAlgebra.jl` packages.

For example, with $k = 1.0$ and $N = 8$, we can generate the stationary distribution for this system. The code implementation allows us to efficiently compute and visualize how the stationary distribution evolves as we change k and N .

The following observations were made for the system with $k = 1.0$ and $N = 8$:

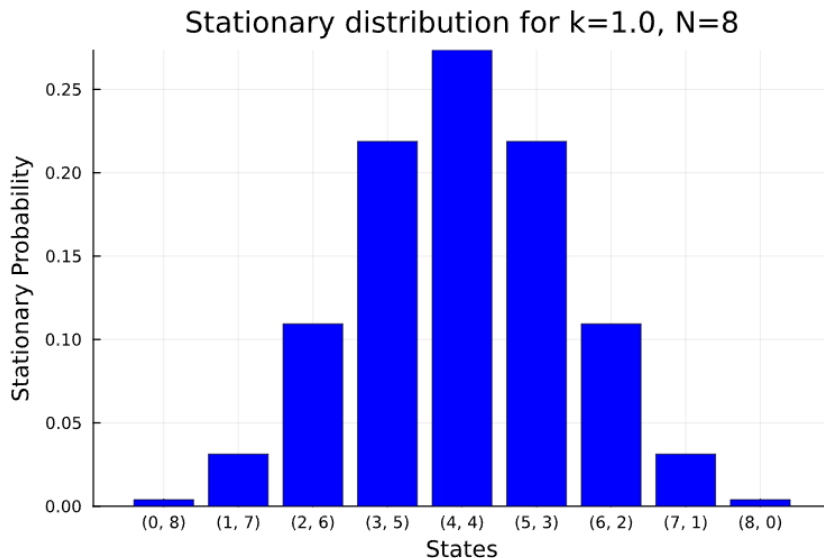


Figure 1: When $k = 1$ in $A \xrightleftharpoons[k=1]{k} B$

In Figure 1 of [3.6.1](#), the system reaches a stationary distribution that exhibits a balanced behavior between the forward and backward reactions. The resulting distribution appears to be either uniform or Gaussian-like, with the peak of the distribution centered around $(4, 4)$, corresponding to an equal distribution of the chemical species A and B . This outcome aligns with the expectation for a system at equilibrium, where the forward and backward rates are balanced, and both species are equally likely to occur in the long term. The uniform or Gaussian-like shape suggests that, for this value of k , the system stabilizes around a central state, reflecting the symmetry and equilibrium of the reaction dynamics.

Now, to further investigate the behavior of the stationary distribution, we will examine two extreme cases for the reaction rate constant k : when $k \ll 1$ and $k \gg 1$.

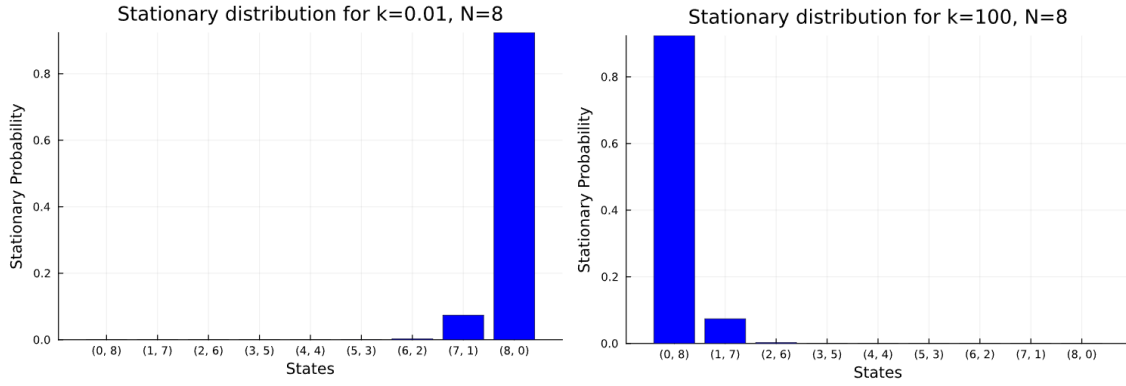


Figure 2: When k in $A \xrightleftharpoons[k]{1} B$ are a) $k=0.01$; b) $k=100$

In Figure 2 of [3.6.2](#) these cases represent scenarios where the forward and backward reactions are significantly imbalanced, and we aim to observe how this imbalance affects the stationary distribution.

Case 1: $k \ll 1$: The backward reaction dominates, favoring states with higher concentrations of A and lower B . The stationary distribution skews towards lower X_A and higher X_B . The computed values of each state are denoted by vector π_1 :

$$\pi_1 = \begin{pmatrix} 0.9234832224823122 \\ 0.07387865779858493 \\ 0.002585753022950477 \\ 5.171506045901179e-5 \\ 6.4643825573933e-7 \\ 5.1715060472606785e-9 \\ 2.585753135800355e-11 \\ 7.387962248013917e-14 \\ 9.319080321307055e-17 \end{pmatrix}$$

Case 2: $k \gg 1$: The forward reaction dominates, favoring states with higher concentrations of B and lower A . The stationary distribution skews towards lower X_B and higher X_A . The computed values of each state are denoted by vector π_2 :

$$\pi_2 = \begin{pmatrix} 9.138003115251386e-17 \\ 7.387905790984913e-14 \\ 2.5857530261719184e-11 \\ 5.171506045575218e-9 \\ 6.46438255736665e-7 \\ 5.17150604590071e-5 \\ 0.0025857530229504793 \\ 0.07387865779858499 \\ 0.9234832224823123 \end{pmatrix}$$

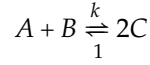
Validation of Approach:

The stationary distributions obtained from our setup align well with theoretical expectations, reinforcing the validity of our approach. These outcomes demonstrate that utilizing the Matrix-Tree Theorem (MCTT) to compute stationary distributions for chemical reaction networks (CRNs) could be a robust and reliable method.

To further solidify this conclusion, we propose analyzing a slightly complex reaction network with three species. By extending the current example to include additional reactions or species, we can test whether the MCTT-based approach consistently produces expected stationary distributions, even for more intricate systems. This will allow us to explore the broader applicability of MCTT in understanding the long-term behavior of CRNs.

3.6.2 Simple Reversible Reaction with Three Species

Consider a reaction network:



The system is governed by two conservation laws:

$$X_B - X_A = M, \quad X_A + X_B + X_C = N,$$

where X_A, X_B, X_C are the concentrations of species A, B, and C, respectively. To ensure non-negative integer concentrations, N and M are chosen such that $N - M$ is even.

State Space:

Given these constraints, the state space will be:

$$\{(i, i + M, N - M - 2i) \mid i = 0, 1, \dots, Z\},$$

where $Z = \frac{N-M}{2}$, and $i, i + M$, and $N - M - 2i$ represent the concentrations of A, B, and C, respectively. The total number of states is $Z + 1$.

Transition rates and Markov Chain

The Laplacian of the transition matrix P is constructed using the following rules:

- **Forward Transitions:** From state i to $i + 1$, corresponding to the reaction $2C \rightarrow A + B$, with rate:

$$P[i + 1, i + 2] = \frac{(N - M - 2i)(N - M - 2i - 1)}{2}.$$

- **Reverse Transitions:** From state $i + 1$ to i , corresponding to the reaction $A + B \rightarrow 2C$, with rate:

$$P[i + 1, i] = k \cdot i \cdot (i + M).$$

- **Diagonal Entries:** Each diagonal entry $P[i + 1, i + 1]$ is the negative sum of the transition rates out of state i :

$$P[i + 1, i + 1] = - \left(\frac{(N - M - 2i)(N - M - 2i - 1)}{2} + k \cdot i \cdot (i + M) \right).$$

Stationary Distribution

Since there is no autocatalytic reaction, our state space is bounded and we can compute the stationary distribution π of this system using MCTT as well. This stationary distribution provides insights into the long-term behavior of the relative concentrations of A, B, and C.

Computational Approach

We chose $N = 12$, $M = 4$ and hence we have 5 states. We took different values of $k = 1, 0.1$ and 100, we have

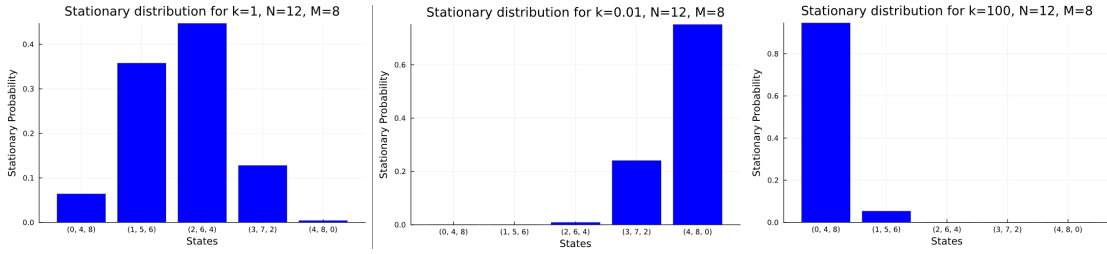


Figure 3: When k in $A + B \xrightleftharpoons[k_1]{k} 2C$ are a) $k=1.0$; b) $k=0.01$; c) $k=100.0$

- When $k = 1.0$, the forward and backward reaction rates are comparable in magnitude. The stationary distribution is spread out, with noticeable probabilities across all states in the reaction network. The balance between A, B, and C is maintained, as neither reaction direction dominates.
- With $k = 0.01$, the forward reaction rate is significantly smaller than the backward rate. This implies that the reaction $A + B \rightarrow 2C$ occurs at a much slower pace than the reverse reaction $2C \rightarrow A + B$. As a result, the stationary distribution becomes concentrated around states where C has higher concentrations, reflecting the dominance of the reverse reaction.
- For $k = 100.0$, the forward reaction $A + B \rightarrow 2C$ is extremely fast compared to the reverse reaction. This drives the system toward states with higher concentrations of C. The stationary distribution becomes heavily biased toward states where C dominates, with negligible probabilities for states where A and B are prevalent.

This further supports the robustness of the approach and motivates its application to larger networks with multiple reactions and species.

3.7 Remark

Through our analysis, we confirmed that MCTT performs well on simple CRNs, accurately predicting the stationary distribution. This validation establishes MCTT as a reliable tool for studying the equilibrium properties of CRNs. Furthermore, the method's success in simple systems paves the way for its application to more complex CRNs, where it can handle a broader range of reaction dynamics and species interactions.

The code used in our experiments is written in Julia and can be found in [\[15\]](#) under Chapter 3.

Chapter 4: Verification of MCTT

We focus on verifying the stationary distribution computed using the Markov Chain Tree Theorem (MCTT) by comparing it against known benchmark method Gillespie available for computing the steady states of a CRN. It is also particularly important to ensure that the stationary distributions obtained from our approach match those derived from well-known stationary distribution formula Product-form stationary distribution for specific CRNs. By conducting such verification, we aim to establish confidence in the robustness of MCTT as a reliable tool for analyzing CRNs across various systems and contexts.

4.1 Verification of MCTT by Product-form Stationary Distribution

We aim to verify the stationary distribution obtained through the MCTT by comparing it with the results from a well-established approach in stochastic chemical reaction network analysis. From [4], we consider examples where chemical reaction systems are modeled with mass-action kinetics and it is proven that a product-form stationary distribution exists for each closed, irreducible subset of the state space, given that the analogous deterministic system with mass-action kinetics admits a complex balanced equilibrium.

To demonstrate the validity of our MCTT-based approach, we will apply it to an example provided in the paper. By comparing the stationary distribution computed from MCTT with the product-form stationary distribution derived from the analytical approach in the paper, we aim to confirm that both methods yield identical results in the case of this particular system.

Example: $A \xrightleftharpoons[k]{1} B$

We will take the *Example 5.1* from [4], we replace species S_1, S_2 by A, B . The product-form stationary distribution for the system is :

$$\pi(x = (x_1, x_2)) = M \frac{c_1^{x_1} c_2^{x_2}}{x_1! x_2!}$$

where

$$M = \frac{1}{\sum_{x \in \Omega_1} \frac{c_1^{x_1} c_2^{x_2}}{x_1! x_2!}}$$

$$c = (c_1, c_2) = \left(\frac{1}{1+k}, \frac{k}{1+k} \right)$$

is the equilibrium condition that satisfies complex balanced condition. See ?? for more. Our state space Ω_1 will have the same states as in 3.6.1

Verification:

We found that **for every value of k , our π_{MCTT} (obtained from MCTT) and $\pi_{ProductForm}$ (obtained via product-form) were actually same.** The table 1 shows the measure between these two stationary distributions given by $D_{KL}(\pi_{ProductForm} \parallel \pi_{MCTT})(k)$ at different values of k :

k	$D_{KL}(\pi_{ProductForm} \parallel \pi_{MCTT})(k)$
0.01	-8.86426×10^{-18}
0.1	-7.81242×10^{-17}
1.0	-4.33681×10^{-17}
10.0	6.63053×10^{-17}
100.0	-6.75823×10^{-18}

Table 1: Comparison of $\pi_{ProductForm}$ and π_{MCTT} for different values of k

This comparison will serve as a rigorous validation of the MCTT method, ensuring that it produces accurate stationary distributions consistent with known theoretical results for CRNs under mass-action kinetics.

4.2 Verification of MCTT by Gillespie Algorithm

The *Gillespie algorithm*, a stochastic simulation method, provides exact samples from the trajectory of a continuous-time Markov chain (CTMC) that models the chemical reaction system. The basic idea behind the Gillespie algorithm is to **simulate the time evolution of a system of chemical reactions by drawing random events from the appropriate probability distributions**, considering both the propensity functions of the reactions and the time between events. This algorithm is often used to simulate and analyze reaction networks with complex stochastic dynamics, and it has been proven to provide exact realizations of the underlying CTMC.

By comparing the stationary distributions of the chemical reaction network obtained from the MCTT method with those generated from the Gillespie algorithm, we aim to validate the accuracy and correctness of the MCTT approach. Specifically, we will simulate same reaction network using both the MCTT and Gillespie methods, and then compare their stationary distributions to ensure they match.

4.2.1 Computational Approach

We first setup a chemical reaction network, the same network and its parameters are used for both MCTT and Gillespie simulations, ensuring a consistent comparison. We first compute the stationary distribution using *Markov Chain Tree Theorem*(MCTT) and denote it as π_{MCTT} . The reaction system is now simulated using the Gillespie algorithm. During computation, we define an initial state u_0 , representing the initial concentrations of the chemical species, and then evolve the system over time. We setup `DiscreteProblem` that represents the chemical reaction system defined by the set of reactions, the `JumpProblem` accounts for the stochastic nature of the system. We use `Direct` method of the Gillespie algorithm to update the system’s state. The `SSAS stepper()` is used to perform the stochastic simulation algorithm (SSA) step by step, and the final state of the system is recorded after a fixed simulation time.

This final state is the steady state of the chemical reaction network after one simulation. We perform such simulations ranging from 700 to 10000 times. The final states from all simulations are collected into a dataset, containing the states of the system. To determine the stationary distribution, we count the occurrences of each state. The stationary distribution $\pi_{Gillespie}$ is then normalized by dividing each state count by the total number of final states.

To measure the difference between the stationary distributions π_{MCTT} and $\pi_{\text{Gillespie}}$, we compute the Kullback-Leibler (KL) divergence:

$$D_{KL}(\pi_{\text{MCTT}} \parallel \pi_{\text{Gillespie}})(n)$$

where n represents the number of simulations. This metric allows us to quantify how closely the distribution obtained from the Gillespie algorithm approximates the one derived from the MCTT method.

By increasing the number of simulations n , we observe how the KL divergence converges, confirming the consistency between the two approaches. This comparison will serve as a more robust verification of the MCTT method’s ability to compute accurate stationary distributions for chemical reaction networks.

4.2.2 Example 1: Simple Reversible CRN $A \xrightleftharpoons[k]{k} B$

We performed computations for the reaction network $A \leftrightarrow B$ with reaction rate $k = 1.0$ and $N = 8$ using MCTT. Then, using the Gillespie algorithm, we generated stationary distributions $\pi_{\text{Gillespie}}$ for numbers of simulations, n . The divergence $D_{KL}(\pi_{\text{Gillespie}} \parallel \pi_{\text{MCTT}})(n)$ was computed for each n , and the results demonstrate the following:

- As the number of simulations n increases, the divergence $D_{KL}(\pi_{\text{Gillespie}} \parallel \pi_{\text{MCTT}})$ decreases, verifying the convergence of $\pi_{\text{Gillespie}}$ to π_{MCTT} .
- **Convergence Proportional to $\frac{1}{\sqrt{n}}$:** The observed rate of convergence of the divergence is approximately proportional to $\frac{1}{\sqrt{n}}$. To confirm this, we plotted the divergence values against n alongside a curve proportional to $\frac{1}{\sqrt{n}}$, given by $\frac{M}{\sqrt{n}}$, where M is a proportionality constant.

The graph in Figure 4.2.2 clearly demonstrates that the divergence follows the $\frac{1}{\sqrt{n}}$ scaling, providing further evidence of the consistency and reliability of the MCTT method in accurately computing stationary distributions.

We now aim to extend our verification to more complex and biologically relevant chemical reaction networks (CRNs). These CRNs are commonly encountered in biological systems and serve as well-established models for studying biochemical processes. By applying the MCTT method to such systems and comparing the results to those obtained using established algorithms, we seek to affirm further the robustness and applicability of MCTT in diverse scenarios.

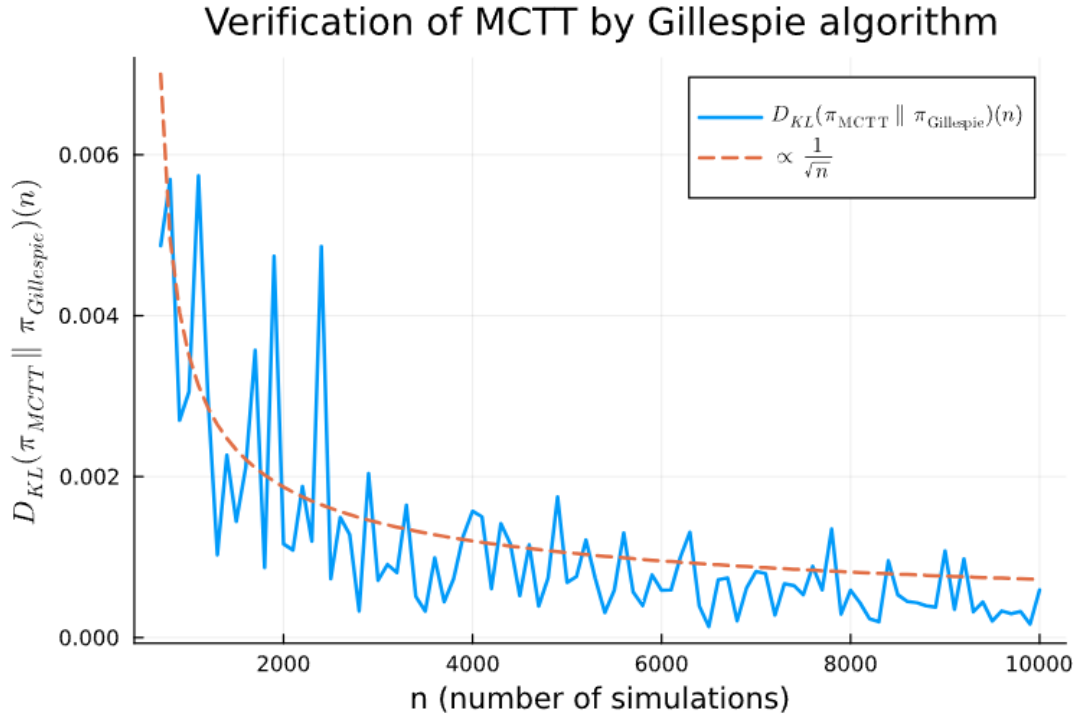
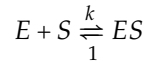


Figure 4: Gillespie Verification of MCTT for $A \xrightleftharpoons[k]{k} B$

4.2.3 Example 2: Enzyme-Catalytic Reaction

We now consider a classic reaction central to the enzyme-catalytic reaction mechanism, specifically the enzyme-substrate binding step. This step forms the foundation of the widely known Michaelis-Menten kinetics, a model extensively used to describe enzymatic reactions. The reaction is represented as:



In the context of enzyme-catalytic reactions: E stands for the enzyme, which is a protein that catalyzes the reaction. S stands for the substrate, which is the molecule upon which the enzyme acts. ES stands for the enzyme-substrate complex, which forms temporarily when the enzyme binds to the substrate during the reaction.

This intermediate complex is crucial in the catalytic process, as it facilitates the conversion of the substrate into the product(s).

This CRN obeys the following conservation laws:

$$S - E = M, E + S + 2ES = N$$

Using these, the state space is parameterized as: $(i, i + M, Z - i)$ where $i = 0, 1, \dots, Z$ with $Z = \frac{N-M}{2}$. To ensure the state space is defined over integers, we choose N and M such that Z is an even number.

For this example, we set $k = 0.3$, $N = 12$ and $M = 4$ to illustrate the process. First, we compute the stationary distribution using MCTT, denoted as π_{MCTT} . Next, we calculate the stationary distribution $\pi_{Gillespie}$ using the Gillespie algorithm with number of simulations n , following the same procedure as in Example 1 [4.2.2](#). The observations for this example align with those of Example 1, $D_{KL}(\pi_{MCTT} \parallel \pi_{Gillespie})(n)$ converge as n increases, demonstrating the accuracy and consistency of the MCTT method. The convergence is proportional to $\frac{1}{\sqrt{n}}$. This can be seen in Figure [4.2.3](#).

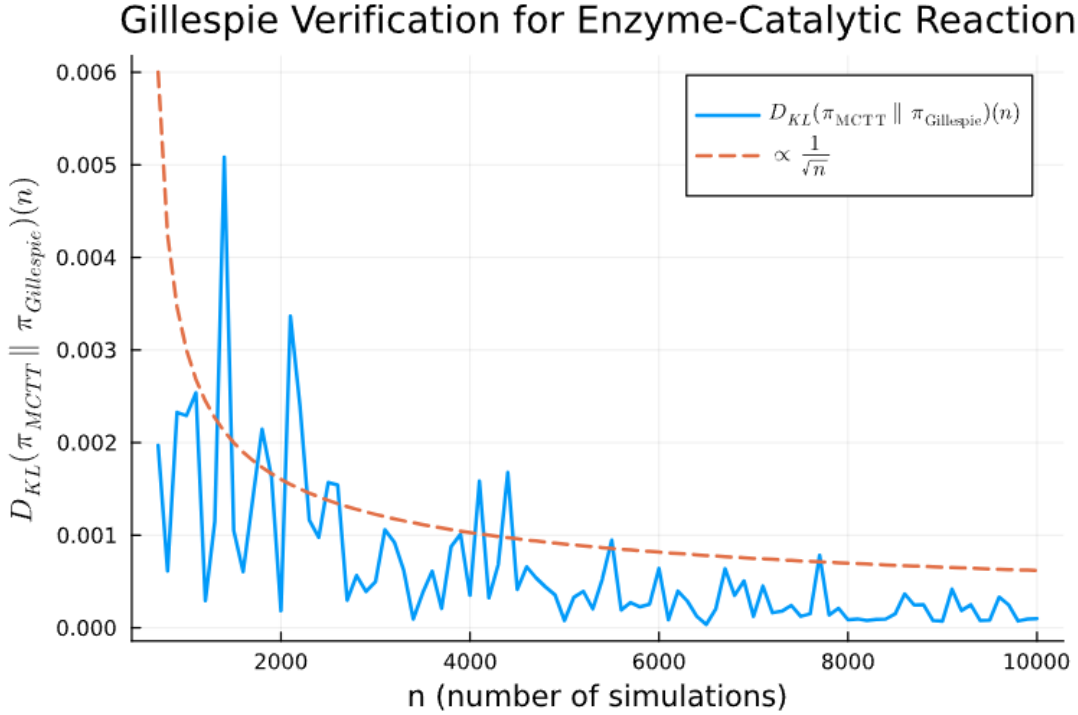


Figure 5: Gillespie Verification of MCTT for Enzyme-Catalytic Reaction

4.2.4 Remark

With the verification achieved for large values of n , One can also sense that while the Gillespie algorithm provides a stochastic method to compute stationary distributions, it can be computationally intensive sometimes. This is because Gillespie relies on simulating numerous trajectories and then estimating the steady-state probabilities of each state. In contrast, the MCTT approach offers a direct method to compute the stationary distribution for at least **small CRNs**, bypassing the need for extensive simulations and thus reducing computational complexity. While the precise time complexity of MCTT for **Large CRNs** remains to be analyzed, it is clear that the method simplifies the process significantly compared to Gillespie at some extent.

These observations suggest that MCTT is a promising approach for efficiently dealing with chemical reaction networks, particularly when computational efficiency is a priority.

The code used in our experiments is written in Julia and can be found in [\[15\]](#) under Chapter 4.

Chapter 5: Learning Using CRNs

In the previous chapters, we explored the computation and verification of stationary distributions for chemical reaction networks (CRNs) using the Markov Chain Tree Theorem (MCTT) and validated the results through comparisons with established techniques such as product-form stationary distributions and the Gillespie algorithm. While this provided a solid foundation for analyzing CRNs, a natural next step is to investigate whether we can reverse the process—learning the parameters governing a CRN from observed distributions.

The ability to infer parameters, such as reaction rates, from experimental data is fundamental to understanding the underlying dynamics of CRNs. This is particularly relevant in systems biology, where experimental measurements often provide the steady-state distributions of molecular species, and the task is to identify the reaction parameters that could reproduce these distributions.

In this chapter, we aim to address this challenge. Specifically, we will explore methodologies for learning the parameters of a CRN given its stationary distribution. By leveraging techniques from optimization, statistical inference, and computational tools, we will investigate how well the parameters can be estimated under different scenarios.

5.1 Learning Rule using MCTT

We aim to develop a learning rule for chemical reaction networks (CRNs) using the Markov Chain Tree Theorem (MCTT). Before applying MCTT, it is essential to confirm whether the state space of the system is bounded. Considering that CRNs induce a continuous-time Markov chain (CTMC) and adhere to conservation laws, along with the absence of reactions involving auto-catalysis or birth-death dynamics, our conjecture 3.4 supports the boundedness of the state space.

By restricting the concentrations of species to integer values, we effectively confine the system to a finite, discrete state space defined on an integer lattice. This bounded state space allows the application of MCTT to compute the stationary distribution, denoted by π_{MCTT} , as a function of k , where k represents the parameters (rate constants in the case of CRNs).

Our goal is to identify the parameters k that minimize the difference between the true data distribution, denoted as π_{true_data} or T (depending on the example), and $\pi_{MCTT}(k)$. This can be formalized by setting up a loss function:

$$f(k) = D_{KL}(T \parallel \pi_{MCTT}(k))$$

Here, f serves as our objective function for the optimization process. For a system with $|\Omega|$ number of states, the objective function can be expressed explicitly as:

$$\text{objective function} = f(k) = \sum_{i=1}^{|\Omega|} T(i) \log \frac{T(i)}{\pi_{MCTT}(i)k} \quad (2)$$

The primary aim is to **minimize this objective function to determine the optimal parameters**. Since minimizing the KL divergence $D_{KL}(T \parallel \pi_{MCTT})$ is equivalent to maximizing the expected log-likelihood of the data under the model π_{MCTT} 13. So, we can also focus on maximizing log-likelihood if we are given data points of a target distribution. Section 5.2 discusses the application of the *argmin* technique for optimization. Section 5.3 explores the use of automatic

differentiation (AD) and gradient descent to minimize the loss function. Section 5.5 provides a detailed discussion on deriving closed-form analytical gradients for a given target dataset.

5.2 Learning using *argmin* method

We will revisit the CRNs discussed in the previous chapters and attempt to recover their rate constants through a data-driven approach. The idea is to determine whether we can accurately infer these parameters using optimization techniques, given the stationary distributions obtained earlier as our target data.

The process involves formulating the problem as an optimization task, where the goal is to minimize the divergence between the observed distribution (e.g., stationary distributions computed using MCTT) and the distribution generated by the model with adjustable rate parameters. Specifically, we will use the **argmin** method, which involves finding the rate parameters that minimize a defined error metric, such as the Kullback-Leibler divergence.

This approach allows us to validate the learning framework in a controlled setting with known rate parameters. By starting with networks for which we already know the rates, we can ensure the **simplicity** of the learning task and evaluate the effectiveness of the chosen method.

5.2.1 Using *argmin* on $A \xrightleftharpoons[k]{k} B$

We consider the CRN defined in 3.6.1 $A \xrightleftharpoons[k]{k} B$ with a rate constant k . We aim to recover the value of k by employing the **argmin-based learning method**. The process begins by generating a stationary distribution for a randomly chosen value of k , say $k = 10$. We will put $N = 8$ and the state space will be same as in 3.6.1. This distribution will serve as our "**true data**" π_{true} for the purpose of this demonstration. Next, we compute stationary distributions for a range of k values spanning 0.01 to 100. For each value of k , we calculate the Kullback-Leibler (KL) divergence, $D_{KL}(\pi_{true} \parallel \pi_{MCTT}(k))$, where $\pi_{MCTT}(k)$ represents the stationary distribution generated for a particular k using MCTT for this CRN and will be our "**model data**".

By plotting D_{KL} as a function of k , we aim to determine the point at which the divergence minimizes. This optimization problem is then solved using the **argmin method**, which identifies the value of k that minimizes the divergence:

$$\hat{k} = \arg \min_k D_{KL}(\pi_{true} \parallel \pi_{MCTT}(k)).$$

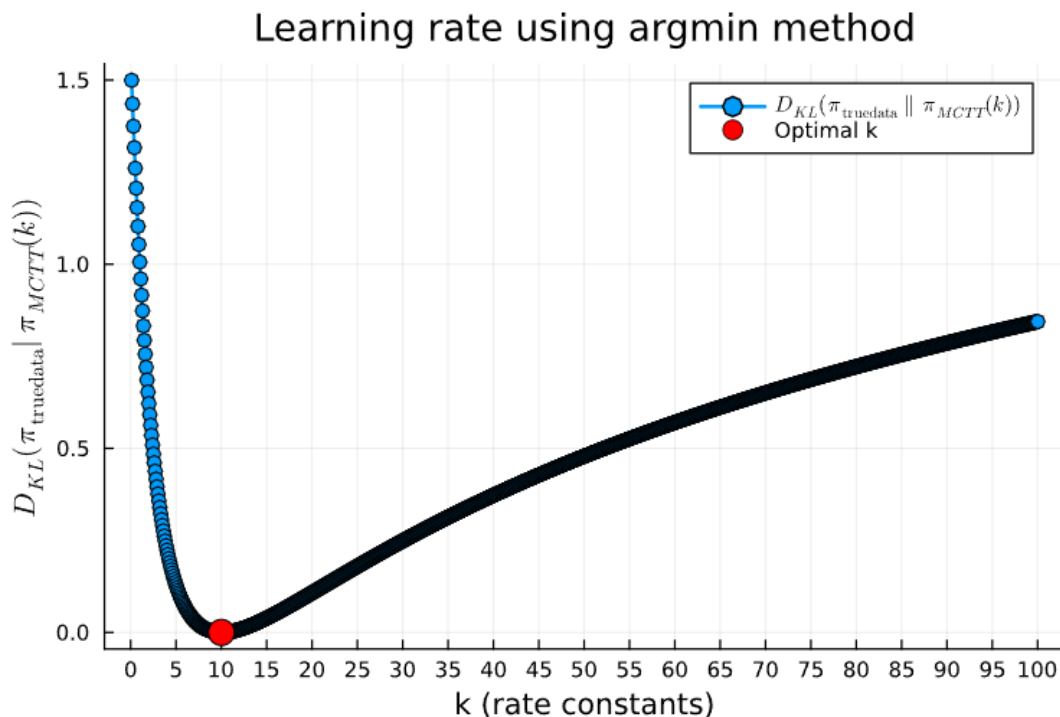


Figure 6: *argmin* method for $A \xrightleftharpoons[k]{k} B$

From the graph in Figure 6 of 5.2.1, it is evident that the *argmin* method successfully identifies the optimal k that aligns with our true data. The graph shows a clear minimum at the value of k corresponding to the best match between the generated stationary distribution and the true stationary distribution. This minimum indicates that the method can effectively recover the parameter k used to generate the true data.

k from a Poisson Distribution:

We assumed that the true rate constant k was known. One can also introduce this problem by asking why did we choose k between 0.01 and 100 in our model data while computing Optimal k . As in many real scenarios, we may not have prior knowledge of the rate constants involved in a given chemical reaction network. To explore this, we now consider the case where the rate constant k is randomly generated from a Poisson distribution. Specifically, we generate k from $Poisson(n)$, where n is a real number that controls the distribution's mean.

The procedure remains similar to our earlier method. For each randomly generated value of k , we compute the stationary distribution $\pi_{MCTT}(k)$ and then apply the *argmin* method to minimize the divergence D_{KL} between the generated stationary distribution and the true data π_{true} . By doing so, we aim to find the value of k that best matches the true distribution.

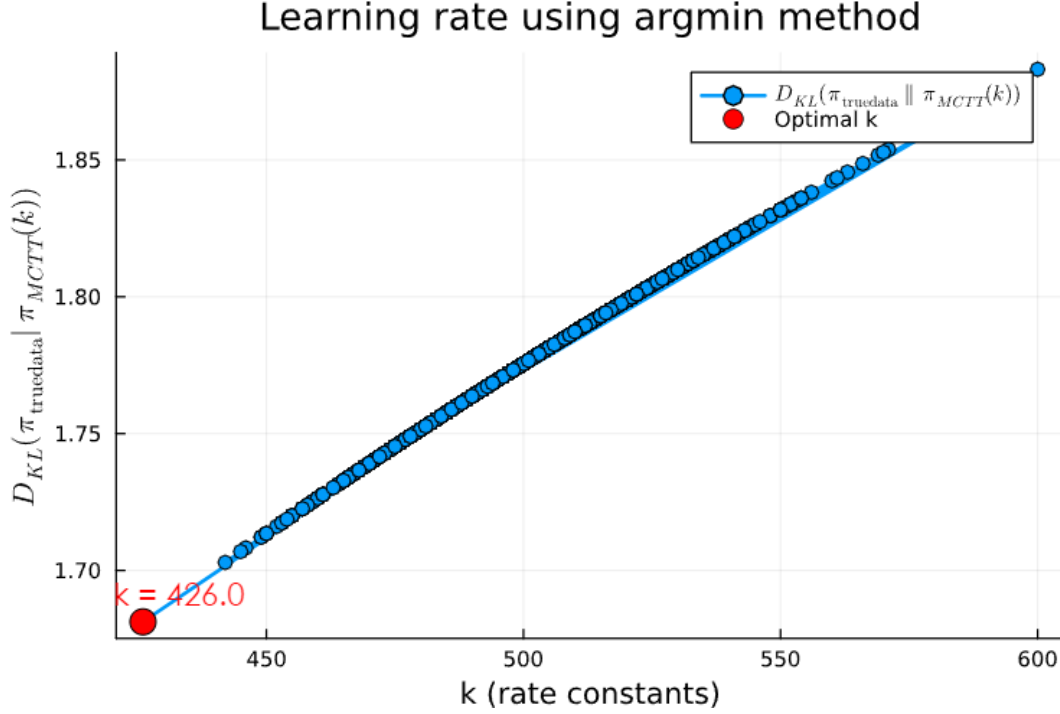


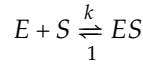
Figure 7: *argmin* with k generated from $Poisson(500)$ in model data

For this CRN, we generated k from $Poisson(500)$ and try to compute the best Optimal k and the best possible divergence. From Figure 7 in 5.2.1 we got $k = 426.0$ to be the best optimal k with $D_{KL} \approx 1.69$. This divergence is notably high, indicating that the estimated k does not closely match the true distribution. An important observation here is that, due to the randomness in the $Poisson(500)$ sampling process, the value of k obtained varies with each search. While this approach provides some insight into the parameter space, the resulting k often fails to meet expectations, as evidenced by the relatively poor divergence values.

This limitation suggests the need for a more systematic and deterministic optimization method. One such approach is gradient descent, which iteratively adjusts k in the direction that minimizes D_{KL} . Gradient descent can provide a more reliable path to the optimal argument, enabling us to better capture the true rate constant k of the system. This motivates us to explore gradient-based methods in the later sections.

5.2.2 Using *argmin* on Enzyme-Catalysis: $E + S \xrightleftharpoons[k_1]{k} ES$

We follow the same approach for our CRN discussed in 4.2.3. We will take $N = 12$, $M = 4$ and $k = 10.0$ to illustrate.



We will first generate k for model data $\pi_{MCTT}(k)$ using k from 0.01 to 100.0 and then we compute the optimal k . From the Figure in 5.2.2, we successfully computed the optimal k as we can see it is

≈ 10 . Now, we will try to take the sampling way in this example also to make our realization from [5.2.1](#) concrete. So, we generate k from $Poisson(500)$ and found the similar conclusion in [Figure 5.2.1](#)

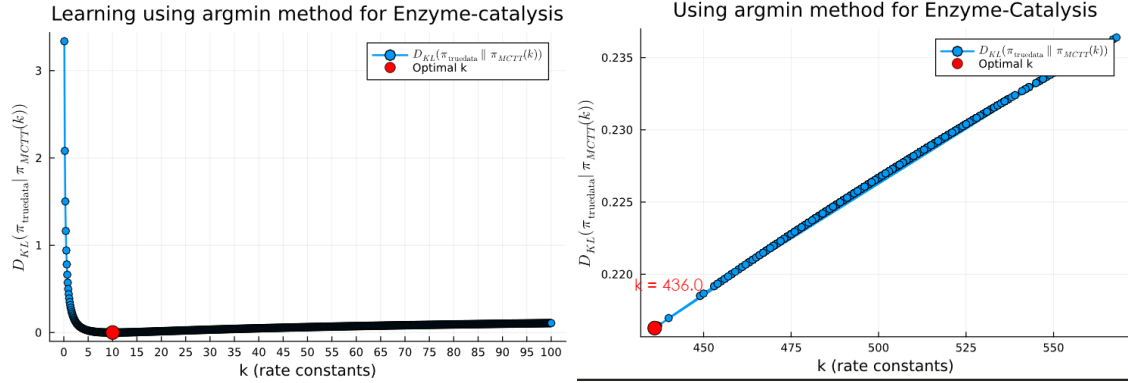


Figure 8: Learning rates using *argmin* method on Enzyme-Catalysis: a) $k = 0.01 : 0.1 : 100$ b) $k \sim Poisson(500)$

From b in Figure 8 of [5.2.2](#), we got best possible $k = 436.00$ with $D_{KL} \approx 0.22$ which was expected to be far away from the actual k .

Using the *argmin* method, we observed similar outcomes for enzyme-catalysis where the accuracy of the estimated rate constant k was dependent on prior knowledge about the parameter space. Without such prior knowledge or an informed starting range, the method often led to notable errors or suboptimal results, and also it could be computationally expensive if we directly consider a very large parameter space.

5.2.3 Limitations

While the *argmin* method offers a straightforward approach to estimating rate constants, it comes with significant limitations. One major challenge is the reliance on preliminary information about the rate parameter. If this initial information is unavailable, incomplete, or inaccurate, the method may yield results that deviate substantially from the true parameters of the system. This was evident in the case where k was randomly sampled from a $Poisson(500)$ distribution; the estimated optimal k and the corresponding D_{KL} were far from satisfactory. Furthermore, the random sampling nature of this method introduces inconsistency, as the outcomes vary with each run. This lack of robustness can hinder its utility, particularly for systems where accurate parameter estimates are critical. The results demonstrate the method's sensitivity to the choice of initial values or sampling distributions, which may not always align with the true dynamics of the system.

5.3 Learning using *gradient-descent* with Automatic Differentiation

The observations from *argmin* method highlight the need for more robust approaches that can navigate the parameter space more effectively and minimize reliance on prior assumptions. Gradient descent offers a systematic way to iteratively refine estimates by moving in the direction of decreasing divergence. This makes it a promising alternative for identifying optimal parameters with greater accuracy and consistency. A critical aspect of implementing gradient descent effectively is the computation of derivatives, which indicate the direction and rate of change of the loss

function with respect to the parameters. While derivatives can be calculated manually or numerically, these methods may become cumbersome or imprecise for complex systems like chemical reaction networks (CRNs). This is where **automatic differentiation (AD)** comes into play.

5.3.1 Automatic Differentiation (AD)

Automatic differentiation is a computational technique that evaluates derivatives of functions accurately and efficiently by breaking down calculations into elementary operations [6]. Unlike numerical differentiation, which suffers from approximation errors, and symbolic differentiation, which can produce unwieldy expressions, AD combines the benefits of both. It computes gradients to machine precision while remaining computationally efficient, making it especially suited for optimization problems involving high-dimensional or complex models.

We employ the `ForwardDiff.jl` package from Julia, which utilizes automatic differentiation to streamline gradient-based optimization. Our approach involves defining the D_{KL} with a list of parameters as its arguments and providing an initial guess for the parameters.

In our computation, the initial parameter values are chosen arbitrarily, provided they do not cause issues in the chemical reaction network (CRN). For the simple CRNs considered here, no such constraints arise, making the process straightforward.

Computational Approach:

The function `objective function(param)` computes the divergence between the stationary distribution predicted by the model data and the true data by taking its argument as parameters `param`. The `initial_param` (initial guess) can be any reasonable value, as the gradient descent algorithm will iteratively refine it. We use the `ForwardDiff.gradient` function to calculate the gradient of the `objective function`. We have to update the parameters `new param` by setting a learning rate. We set the tolerance and `max iterations` that will define the stopping criteria.

```

gradient objective = param → ForwardDiff.gradient(objective function, param)

for iteration in max iterations
    grad = gradient objective(param) # Compute gradient
    new param = param - learning rate grad # Update parameters
    if norm(new param - param) < tolerance # Check for convergence
        println("Converged after iteration iterations")
        break
    end

```

The final `new param` will be our optimal parameter. Through this process, we compute the optimal parameters for our true data using gradient descent. The `ForwardDiff` package ensures accurate and efficient gradient computations. This approach overcomes the limitations of the *argmin* method, offering a more systematic and robust solution for learning rate parameters in CRNs.

5.3.2 AD in $A \xrightleftharpoons[k]{k} B$

We will employ the computational approach discussed in [5.3] to our CRN in [5.2.1]. We will take $\text{true_data} = \pi_T = \pi_{MCTT}(10.0)$ i.e. $k=10.0$ and we define objective function $f(k)$ from the equation

(2). We set tolerance = 1e-10 and max iterations = 50000. Since, in this CRN we only have one parameter as k , our param will have only one element. We now initialize our parameter with $\sim \text{Poisson}(500)$ and for a faster convergence we took learning rate = 5.0, we plotted the graph of $f(k)$ versus iterations in Figure 9 [5.3.2](#)

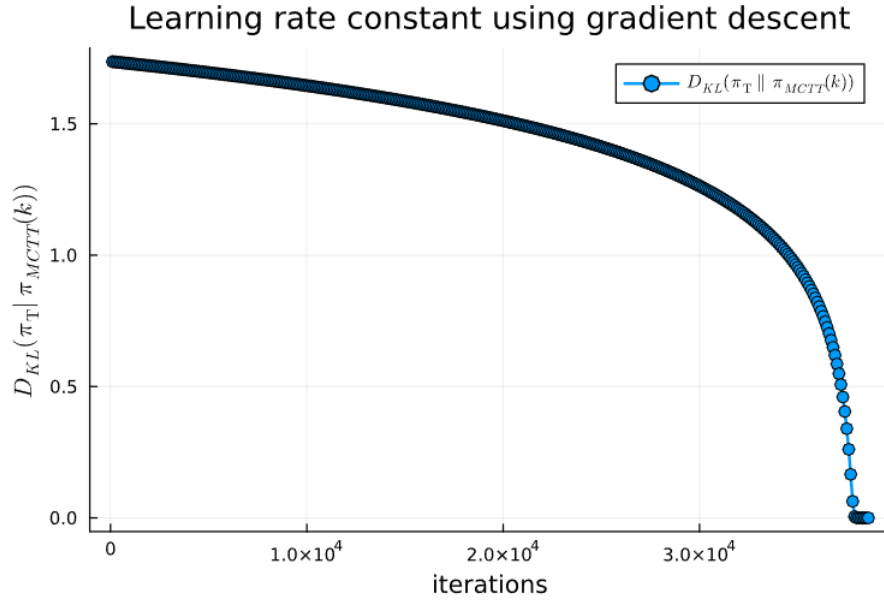


Figure 9: Convergence of $f(k)$ with increasing number of iterations.

We got the output as:

Optimized Parameter: = 10.00 correct upto 10 decimal places

Even though the initial parameter value was chosen randomly from a Poisson distribution, which was far from the true value, the optimization successfully converged to an optimal $k \approx 10.0$, demonstrating the effectiveness of the gradient descent optimization method in finding the optimal parameters, even when starting from an imperfect initial guess.

Note:

Since the initial value of k is randomly generated from a Poisson distribution with a mean of 500, the optimal value for k may vary with each run. However, as demonstrated by our code and computational techniques, it consistently converges to a value near 10.0. While the accuracy can be further improved, the key takeaway should be the **effectiveness of gradient descent with Automatic Differentiation (AD)** in optimizing the model's parameters.

5.3.3 AD in Enzyme-Catalysis: $E + S \xrightleftharpoons[k_1]{k} ES$

For our second example in [5.2.2](#), we applied the same approach by taking k randomly from a $\text{Poisson}(500)$. Despite starting with an initial guess far from the true value, we successfully

achieved an optimal value for $k \approx 10.00$. The convergence can be seen in Figure 10 [5.3.3](#). This result reinforces the effectiveness of using gradient descent with automatic differentiation (AD) as a promising tool for parameter optimization in CRNs.

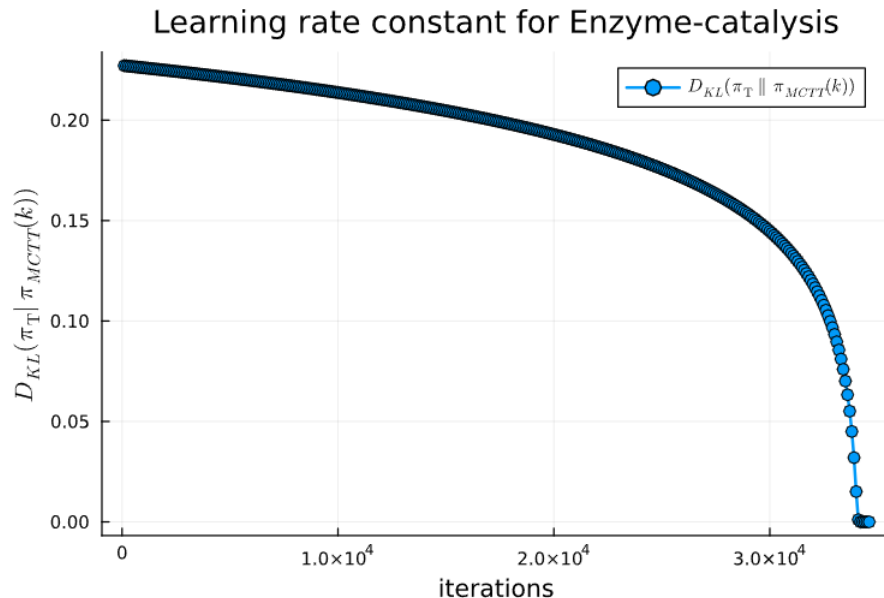


Figure 10: Gradient descent using AD in learning rate constant for Enzyme-catalysis

5.4 Learning from Energy-Based models

We have explored how CRNs can learn their rate parameters from the stationary distributions of known CRN systems. Building on this foundation, we now aim to investigate whether CRNs can learn their rates from probability distributions in a more general setting. To do this, we will explore energy-based models (EBMs), which are a class of probabilistic models where the probability distribution over states is defined through an energy function.

5.4.1 Energy-Based Models

Energy-based models provide a powerful framework for learning in systems with complex dependencies [\[12\]](#). In these models, the probability of a state is given by an exponential function of the negative energy associated with that state. Formally, for a given state x , the probability distribution is defined as:

$$P(x) = \frac{1}{Z} e^{-E(x)}$$

where $E(x)$ is the energy function associated with state x , Z is the partition function, which normalizes the distribution over all possible states (i.e., ensures that the probabilities sum to 1). If Ω is the set of all possible states, we have:

$$Z = \sum_{x \in \Omega} e^{-E(x)}$$

To illustrate the potential of CRNs in learning energy-based models (EBMs), we will first consider a simple EBM where the probability $P(x)$ for each state x can be computed directly where the partition function Z is **interactable**. As this Z is usually interactable when we have a large number of states or we don't have enough information of all states. This simplification allows us to sidestep the computational challenges often associated with normalizing constants in EBMs.

5.4.2 Three Node Energy-Based Model

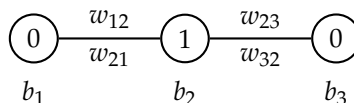
We consider a three-node system, where each node can take a binary state: 0 or 1. Nodes influence the states of their adjacent nodes only, creating local dependencies. Given these constraints, the total number of possible states in the system is $2^3 = 8$. The state space will be:

$$\Omega = \{0,1\}^3 = \{(0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1)\}$$

The set $\{0,1\}^n$ is defined as the Cartesian product of n copies of the set $\{0,1\}$, formally given by:

$$\{0,1\}^n = \{(x_1, x_2, \dots, x_n) \mid x_i \in \{0,1\}, \forall i = 1, 2, \dots, n\}.$$

The graph below represents the three-node model for the state $(0,1,0)$. Each node is labeled with its state, and edges indicate interactions between adjacent nodes.



To define the initial system as **true data**, we assign random biases b_i to each node, which determine their intrinsic tendency toward a state of 1 or 0 and assign random weights w_{ij} to the edges connecting adjacent nodes i, j , which describe the interaction strength between them. The **energy function** of a state $x = (x_1, x_2, x_3) \in \Omega$ is given by:

$$E(x) = - \sum_i b_i x_i - \sum_{i,j} w_{ij} x_i x_j$$

Using definition of Z and $P(x)$ from [5.4.1](#) we calculate the probability distribution and call it as T . For illustration, we take biases $b = [0.5, 0.3, 0.8]$ and weights $W = [0.4, 0.2]$ and computed $P(x)$ for each state and computed the target distribution T that can be seen in Figure 11:

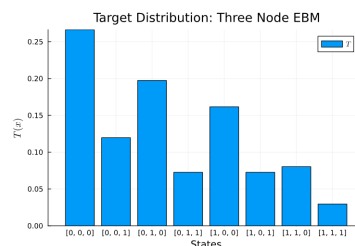
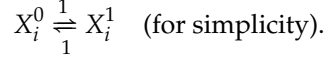


Figure 11: T as Three Node EBM

CRN and Learning CRN parameters for T :

Firstly, we have to construct a chemical reaction network (CRN) whose state space matches that of the EBM. This involves designing species and reactions that align with the structure of the EBM. The state space of the CRN, Ω_{CRN} , should match the state space Ω of the EBM, i.e., $\{0,1\}^3$, representing the binary states of the three nodes. Each node in the EBM is represented by a chemical

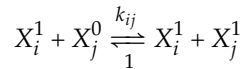
species X_i , which can exist in two states: X_i^0 (off) or X_i^1 (on). To model the tendency of nodes to switch states and the interactions between adjacent nodes, we define the following reactions as: Each node i can independently switch between states 0 and 1:



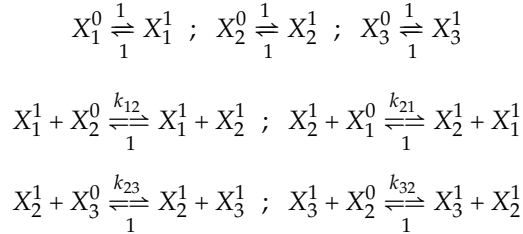
For each node i , as they can be either turned "ON" or "OFF" at a time, so the total concentration remains constant:

$$X_i^0 + X_i^1 = 1.$$

Since the nodes influence the states of their neighbors, we define reversible reactions that describe these interactions. For each pair of adjacent nodes i and j , we have:



These k_{ij} will be our parameters in this CRN. Hence, our CRN will be:



We can see that our CRN has 6 species: $X_1^0, X_1^1, X_2^0, X_2^1, X_3^0, X_3^1$ and we can assume that each node in EBM X_i can be represented by a pair of species (X_i^0, X_i^1) . The three conservation laws are:

$$X_1^0 + X_1^1 = 1 ; \quad X_2^0 + X_2^1 = 1 ; \quad X_3^0 + X_3^1 = 1$$

The state space for our CRN is exactly $\{0, 1\}^3$, which matches the state space of the energy-based model (EBM) Ω . Therefore, we have $\Omega_{CRN} = \Omega$, as we already know CRNs induce CTMC and since it doesn't have any autocatalytic or birth-death reaction, our conjecture 3.4 gives us our state space to be bounded which is obvious in this example, we can apply MCTT to compute the stationary distribution $\pi_{MCTT}(\kappa)$ for different rate constants $\kappa = \{k_{12}, k_{21}, k_{23}, k_{32}\}$. The goal is to optimize the rate constants κ such that the stationary distribution approximates the target distribution T . We define our objective function $f(\kappa)$ from 2:

$$f(\kappa) = D_{KL}(T \| \pi_{MCTT}(\kappa))$$

We apply gradient descent on $f(\kappa)$ in the same way described in 5.3 to compute the gradients of the $f(\kappa)$ with respect to the rate constants and then analyze the behavior of the optimization process by plotting $f(\kappa)$, against the number of iterations during the gradient descent in Figure 12:

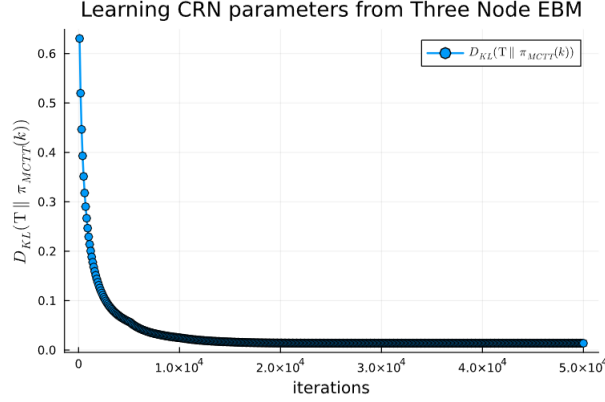


Figure 12: Learning CRN Parameters for Three-Node EBM Using Gradient Descent with Automatic Differentiation

Figure 12 shows that the value of $f(\kappa)$ steadily decreases and tends to converge towards zero as the number of iterations increases. This indicates that the CRN is successfully learning its rate parameters to approximate the target distribution T .

Learning Rate Sensitivity:

While the convergence trend is promising, achieving the exact optimal parameters depends on selecting an appropriate learning rate. A learning rate that is too high may cause instability, while a very small learning rate might slow down convergence. Fine-tuning the learning rate is essential to ensure efficient and accurate optimization.

5.4.3 Significance of Learning CRNs from EBMs

The trend of convergence strongly supports the idea that CRNs can be used to model and learn from probability distributions, even complex energy-based models (EBMs) and are capable of replicating intricate distributions by adjusting their rate constants. The use of **automatic differentiation (AD)** makes gradient-based optimization feasible and accurate, even for complex dynamical systems like CRNs. This success opens the possibility of using CRNs for various machine learning tasks, particularly in modeling stochastic systems and learning from data.

5.4.4 Considerations and Possible Challenges

While the convergence trend observed in the optimization process is promising, achieving the exact optimal parameters necessitates careful tuning of hyperparameters, particularly the learning rate and initial guesses. Expanding this approach to larger state spaces or more complex CRNs introduces several challenges, including the intractability of the partition function in larger energy-based models (EBMs), computational inefficiencies, and optimization robustness. As the state space size and CRN complexity increase, the time and resources required for each gradient descent iteration grow, potentially hindering practical application for larger systems. The computational cost of evaluating the partition function also escalates in EBMs with numerous nodes and interactions, limiting the scalability of the method.

5.5 Learning CRN Parameters from Data

In this section, we aim to extend our approach by learning the rate parameters of a CRN directly from a dataset, which could be empirical, synthetic, or otherwise. Let \mathcal{D} denote the dataset, and our goal is to extract underlying observations, patterns, and feature classifications from it using CRNs.

5.5.1 Construction of CRN

To achieve this, we first need to set up a CRN that induces a continuous-time Markov chain (CTMC), with the constraint that it contains no autocatalytic or birth-death reactions, ensuring that the state space remains bounded. Suppose that, for a given dataset \mathcal{D} , we have such a CRN represented by $CRN = (\mathcal{S}, \mathcal{R}, \kappa)$ [3.1](#)

The complexes of this CRN are denoted by \mathcal{C} , which is a subset of $\mathbb{Z}_{\geq 0}^{|\mathcal{S}|}$. Consequently, the reactions of the CRN will be a subset of $\mathcal{C} \times \mathcal{C}$, and κ is a function that maps each reaction in \mathcal{R} to a non-negative real number representing the corresponding rate constant. This κ will be our set of parameters.

$$\mathcal{C} \subset \mathbb{Z}_{\geq 0}^{|\mathcal{S}|}, \quad \mathcal{R} \subset \mathcal{C} \times \mathcal{C}$$

$$\kappa : \mathcal{R} \rightarrow \mathbb{R}_{\geq 0}$$

From [3.3](#), this CRN induce a CTMC on the state space $\mathbb{Z}_{\geq 0}^{|\mathcal{S}|}$ with transition rates from state s to s' given by $Q(\kappa)$, which can be expressed as:

$$Q(\kappa)(s \rightarrow s') = \sum_{\substack{\mathbf{y} \rightarrow \mathbf{y}' \in \mathcal{R} \\ \mathbf{s} \geq \mathbf{y} \\ \mathbf{s} + \mathbf{y}' - \mathbf{y} = \mathbf{s}'}} \kappa(\mathbf{y} \rightarrow \mathbf{y}') \binom{\mathbf{s}}{\mathbf{y}} \mathbf{y}! \quad (3)$$

Additionally, from [3.3.1](#) the reachability class from an initial state \mathbf{s}_0 will be:

$$\Omega_{\mathbf{s}_0} = \mathcal{C}[\mathbf{s}_0] = \mathbf{s}_0 + \text{span}\{\mathbf{y}' - \mathbf{y} \mid \mathbf{y} \rightarrow \mathbf{y}' \in \mathcal{R}\} \cap \mathbb{Z}_{\geq 0}^{|\mathcal{S}|} \quad (4)$$

The matrix Q (laplacian of transition matrix) will have entries given by:

$$Q_{ss'} = \begin{cases} Q(\kappa)(s \rightarrow s'), & \text{if } s \neq s' \\ - \sum_{s_c \in \{\Omega_{\mathbf{s}_0} \setminus s\}} Q(\kappa)(s \rightarrow s_c), & \text{if } s = s' \end{cases} \quad (5)$$

We characterize the CRN such that Ω_0 is bounded by using Conjecture [3.4](#)

We assume that $\Omega_0 = \Omega \times H$, where H represents potential hidden variables. This setup provides the foundation for learning the rate constants of the CRN directly from the data, enabling us to infer the underlying generative process of the observed patterns in the dataset.

5.5.2 Stationary Distribution and Marginalization

To formalize the model's data representation, we define the stationary distribution of a state s using Theorem 1 [2.4](#). For a state s , using equation [\(1\)](#) the stationary distribution is given by:

$$\pi(s) = \frac{\text{adj}(Q)^{s,s}}{Z} = \frac{\det(Q^{s,s})}{Z}$$

where $Q^{s,s}$ represents the matrix obtained by deleting the s -th row and s -th column of matrix Q , and Z is the partition function.

Since the state s can be expressed as a tuple $s = (w, h)$, where $w \in \Omega$ and $h \in H$, we define the marginal distribution over Ω as:

$$\pi_{\Omega}(w) = \sum_{h \in H} \pi(w, h).$$

We consider $m(s = (w, h)) = \det(Q^{s,s}) = \det(Q^{wh,wh})$. Thus, the joint stationary distribution can be written as:

$$\pi(w, h) = \frac{m(w, h)}{Z},$$

where the marginal determinant is given by:

$$m_{\Omega}(w) = \sum_{h \in H} m(w, h).$$

The partition function Z is defined as:

$$Z = \sum_{(w,h) \in \Omega \times H} m(w, h).$$

5.5.3 Defining Loss-function of Data

Given a target dataset \mathcal{D} , that generates a target distribution T from samples $\{\mathcal{D}_1, \dots, \mathcal{D}_N\}$, where $\mathcal{D}_i = \{v_j\}_{j=1}^n$ represents a batch of n data points v_j . Our objective is to learn the parameters κ of the CRN such that the model distribution π approximates the target distribution T . Using (2), we have our objective function $f(\kappa)$ as function of parameter set κ :

$$\operatorname{argmin}_{\kappa} f(\kappa)$$

or equivalently,

$$\operatorname{argmin}_{\kappa} D_{\text{KL}}(T \| \pi_{\Omega}),$$

We define $f(\kappa)$ as our loss function, we employ gradient descent to minimize it. Using automatic differentiation as described in Section 5.3, we iteratively adjust κ to reach the optimal parameters. For a given batch of datapoints $\{v_i\}_{i=1}^n$, We define its Loss function as:

$$f(\kappa | \{v_i\}_{i=1}^n) = D_{\text{KL}}(T \| \pi_{\Omega}) f(\kappa | \{v_i\}_{i=1}^n) = \sum_{i=0}^n T(v_i) \log \frac{T(v_i)}{\pi_{\Omega}(v_i)} \quad (6)$$

Minimizing this over κ ensures that the model's parameters are adjusted to best fit the observed data.

5.5.4 Computing Gradient

We compute its gradient with respect to the parameters.

$$\frac{\partial f(\kappa, |\{v_i\}_{i=1}^n)}{\partial \kappa} = \frac{\partial}{\partial \kappa} \sum_{i=1}^n T(v_i) \log \frac{T(v_i)}{\pi_{\Omega}(v_i)}$$

$$= \frac{\partial}{\partial \kappa} \sum_{i=1}^n (T(v_i) \log T(v_i) - T(v_i) \log \pi_{\Omega}(v_i))$$

As T for a given batch will remain constant, So:

$$\begin{aligned} &= -\frac{\partial}{\partial \kappa} \sum_{i=1}^n T(v_i) \log \sum_{h \in H} \pi(w, h) \\ &= -\frac{\partial}{\partial \kappa} \sum_{i=1}^n T(v_i) \log \sum_{h \in H} \frac{m(v_i, h)}{Z} \\ &= -\frac{\partial}{\partial \kappa} \sum_{i=1}^n T(v_i) \left(\log \sum_{h \in H} m(v_i, h) - \log Z \right) \\ &= -\frac{\partial}{\partial \kappa} \sum_{i=1}^n T(v_i) \log \sum_{h \in H} m(v_i, h) + \frac{\partial}{\partial \kappa} \sum_{i=1}^n T(v_i) \log Z \end{aligned}$$

Since $\log Z$ will be independent of i and $\sum_{i=1}^n T(v_i) = 1$, so on substituting Z , we get:

$$\begin{aligned} &= -\sum_{i=1}^n T(v_i) \frac{\partial}{\partial \kappa} \log \left(\sum_{h \in H} m(v_i, h) \right) + \frac{\partial}{\partial \kappa} \log \left(\sum_{(w, h) \in \Omega \times H} m(w, h) \right) \\ &= -\sum_{i=1}^n T(v_i) \frac{\partial}{\partial \kappa} \log \left(\sum_{h \in H} \det(Q^{v_i h, v_i h}) \right) + \frac{\partial}{\partial \kappa} \log \left(\sum_{(w, h) \in \Omega \times H} \det(Q^{wh, wh}) \right) \\ \frac{\partial l(\kappa, \{v_i\}_{i=1}^n)}{\partial \kappa} &= -\sum_{i=1}^n T(v_i) \frac{\frac{\partial}{\partial \kappa} \left(\sum_{h \in H} \det(Q^{v_i h, v_i h}) \right)}{\sum_{h \in H} \det(Q^{v_i h, v_i h})} + \frac{\frac{\partial}{\partial \kappa} \left(\sum_{(w, h) \in \Omega \times H} \det(Q^{wh, wh}) \right)}{\sum_{(w, h) \in \Omega \times H} \det(Q^{wh, wh})} \end{aligned} \quad (7)$$

Jacobi's formula:

Jacobi's formula states that for a differentiable matrix function $Q(\kappa)$, the derivative of its determinant with respect to the parameter κ is given by:

$$\frac{\partial}{\partial \kappa} \det(Q(\kappa)) = \det(Q(\kappa)) \cdot \text{tr} \left(Q(\kappa)^{-1} \frac{\partial Q(\kappa)}{\partial \kappa} \right) \quad (8)$$

where $\text{tr}(\cdot)$ is the trace of matrix. Combining equation (7) and (8), we have $\frac{\partial f(\kappa, \{v_i\}_{i=1}^n)}{\partial \kappa}$:

$$= -\sum_{i=1}^n T(v_i) \left(\frac{\sum_{h \in H} \det(Q^{v_i h, v_i h}) \cdot \text{tr} \left((Q^{v_i h, v_i h})^{-1} \frac{\partial Q^{v_i h, v_i h}}{\partial \kappa} \right)}{\sum_{h \in H} \det(Q^{v_i h, v_i h})} \right) + \left(\frac{\sum_{(w, h) \in \Omega \times H} \det(Q^{wh, wh}) \cdot \text{tr} \left((Q^{wh, wh})^{-1} \frac{\partial Q^{wh, wh}}{\partial \kappa} \right)}{\sum_{(w, h) \in \Omega \times H} \det(Q^{wh, wh})} \right)$$

The term

$$\frac{\sum_{h \in H} \det(Q^{v_i h, v_i h}) \cdot \text{tr} \left((Q^{v_i h, v_i h})^{-1} \frac{\partial Q^{v_i h, v_i h}}{\partial \kappa} \right)}{\sum_{h \in H} \det(Q^{v_i h, v_i h})}$$

is a weighted sum of the quantities $\text{tr} \left((Q^{v_i h, v_i h})^{-1} \frac{\partial Q^{v_i h, v_i h}}{\partial \kappa} \right)$, with the weights given by $\det(Q^{v_i h, v_i h})$. This is equivalent to the expected value of the trace term under the distribution of h conditioned on the data point v_i . Specifically, this can be expressed as:

$$E_{\pi(h|v_i)} \left[\text{tr} \left((Q^{v_i h, v_i h})^{-1} \frac{\partial Q^{v_i h, v_i h}}{\partial \kappa} \right) \right]$$

where $\pi(h|v_i)$ is the conditional probability distribution of h given v_i , and the expectation is taken over this distribution.

Similarly, the second term

$$\left(\frac{\sum_{(w,h) \in \Omega \times H} \det(Q^{wh, wh}) \cdot \text{tr} \left((Q^{wh, wh})^{-1} \frac{\partial Q^{wh, wh}}{\partial \kappa} \right)}{\sum_{(w,h) \in \Omega \times H} \det(Q^{wh, wh})} \right)$$

can also be interpreted as an expected value with respect to the joint distribution over (w, h) , with weights given by $\det(Q^{wh, wh})$. This term is the expected value of the trace term with respect to the joint distribution of (w, h) , which can be written as:

$$E_{\pi(w,h)} \left[\text{tr} \left((Q^{wh, wh})^{-1} \frac{\partial Q^{wh, wh}}{\partial \kappa} \right) \right]$$

Thus, the gradient of the loss function can be written more compactly as:

$$-\frac{\partial f(\kappa | \{v_i\}_{i=1}^n)}{\partial \kappa} = \sum_{i=1}^n T(v_i) \mathbb{E}_{\pi(h|v_i)} \left[\text{tr} \left((Q^{v_i h, v_i h})^{-1} \frac{\partial Q^{v_i h, v_i h}}{\partial \kappa} \right) \right] - \mathbb{E}_{\pi(w,h)} \left[\text{tr} \left((Q^{wh, wh})^{-1} \frac{\partial Q^{wh, wh}}{\partial \kappa} \right) \right] \quad (9)$$

Equation (9) expresses the gradient as the difference between two expectations, one conditioned on v_i and the other on the joint distribution over (w, h) .

The *first term* represents the data-driven influence on the gradient, meaning it reflects how well the model fits the actual data points. The *second term* is the model-driven influence on the gradient, which reflects how the parameters should be adjusted to match the joint distribution of the model itself.

5.5.5 Sampling-based computation of Gradient

Here, each datapoint v_i corresponds to an observable species and has a dimensionality of $|w|$, we begin by initializing the parameter set κ . From equation (9), the loss function consists of two terms, representing the data-driven influence and the model-driven influence on the gradient. For the data-driven influence, we focus on sampling the hidden state h from the conditional distribution

$\pi(h|v_i)$. This is done by **clamping** all the observable species w , which effectively disables the reactions that could alter the configuration of observable species. We initialize the parameters that do not influence observable species and use the **Gillespie algorithm** to obtain the configuration of the hidden species h . This process generates a sample $h_{\text{constructed}} \sim \pi(h|v_i)$. This is then used as h , alongside the observable data point v_i , to compute the first term of the gradient. This process is repeated for all n datapoints.

For the model-driven influence, computing the joint distribution $\pi(w, h)$ becomes intractable for large models. To overcome this, we first construct $h_{\text{constructed}}$ from a random v_i , then **clamp** the reactions that alter the configuration of the hidden species. The **Gillespie algorithm** is used again to compute the configuration of the observable species, denoted as $w_{\text{constructed}}$. This constructed pair $(w_{\text{constructed}}, h_{\text{constructed}})$ is used to compute the second term of the gradient. This two-step approach allows us to efficiently approximate the gradient of the log-likelihood without directly calculating the intractable joint distribution.

5.5.6 Updating Parameter and Convergence

After computing the gradient, $\frac{\partial f(\kappa|\{v_i\}_{i=1}^n)}{\partial \kappa}$, using equation 9 and the sampling procedure outlined in 5.5.5, we proceed with updating the parameters. The learning process follows a gradient descent approach where the parameters κ are updated iteratively according to the rule:

$$\kappa := \kappa - \text{learning rate} \times \left(\frac{\partial l(\kappa|\{v_i\}_{i=1}^n)}{\partial \kappa} \right)$$

The update continues until the convergence or stopping criteria are met, which typically include tolerance thresholds for the change in the loss function and a maximum number of iterations. These criteria ensure that the algorithm converges to an optimal set of parameters that best approximate the data distribution.

Once the parameters have converged, the learned chemical reaction network (CRN) can be used to gain insights into the data. The CRN, with its optimized parameters, provides a powerful tool for modeling complex systems and understanding the underlying dynamics from the observed data.

5.6 Example using Learning Rule

To demonstrate the effectiveness of the proposed learning rule 9, we first consider the 3-Node Boltzmann Machine, defined in Figure 11 in 5.4.2 as the target distribution. To learn this target distribution, we employ a Taylor Chemical Boltzmann Machine (TCBM) setup [8], wherein three pairs of species are designated as observable, while two pairs are treated as hidden. This structure ensures that the chemical reaction network (CRN) induces a distribution over both observable and hidden states.

The learning rule is applied iteratively, updating the parameter set κ using gradient descent. A learning rate of 0.5 is used, and the process runs for 1000 iterations. Throughout the training, we sample configurations for the observable and hidden species following the methods described earlier to compute the gradient and update the parameters.

After 1000 iterations, the learned distribution achieves good convergence, closely approximating the target distribution. This demonstrates the validity and robustness of the learning approach. Figure 13 provides the visual confirmation of the accuracy of the method.

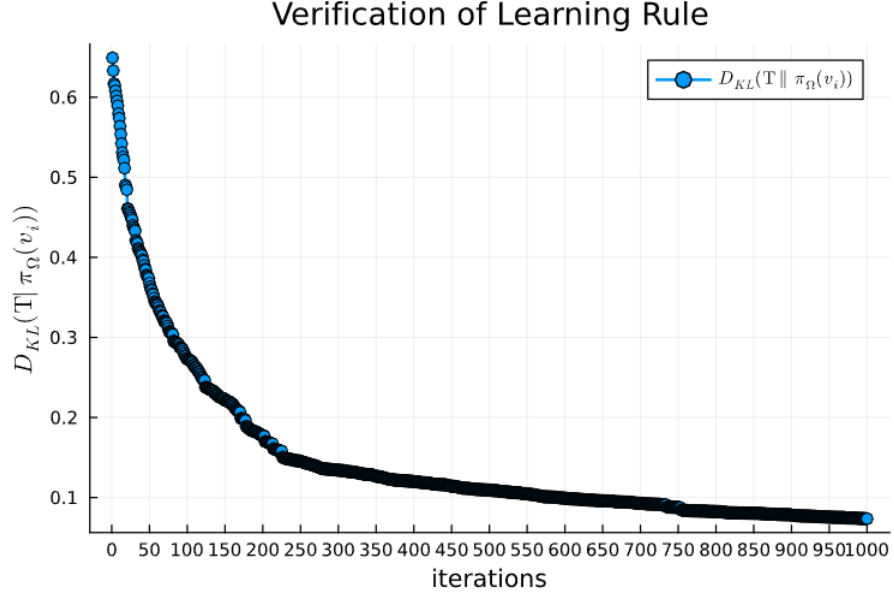


Figure 13: Learning Optimal Rates using Learning Rule defined in Equation (9)

To further validate the accuracy of our learning approach, we overlay the two distributions T and the distribution obtained from CRN setup after learning optimal parameters for a direct visual comparison.

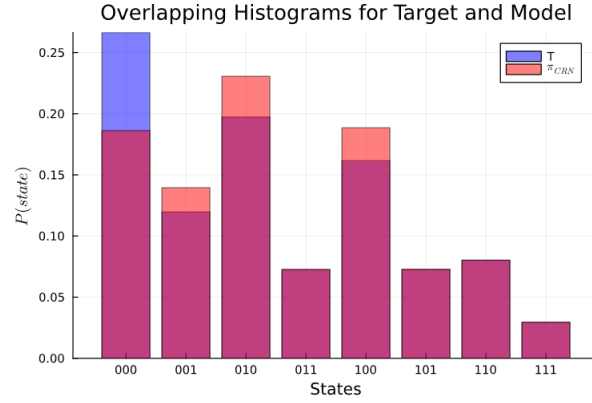


Figure 14: Overlay of Target and Model Distribution with optimized parameters

This overlap provides an intuitive and clear representation of the alignment between the target and learned distributions. Any deviations, if present, become immediately apparent. As seen in the resulting figure, the two distributions show excellent agreement, confirming that the learned CRN parameters successfully replicate the target distribution. This visual confirmation complements the quantitative analysis and demonstrates the robustness and precision of the proposed learning framework.

5.7 Key Challenges with CRN-based Learning

Although we have established a learning rule for parameter estimation from any target dataset, several challenges persist in applying this rule effectively. One key question is whether a reliable Chemical Reaction Network (CRN) exists for a given target dataset. While we cannot guarantee the existence of such a CRN for every dataset, we can characterize the structure of CRNs for certain classes of datasets. For example, in Section 5.4.1, we demonstrated that CRNs can learn reaction rates from energy-based models (EBMs), which can, in turn, be learned through Boltzmann Machines [9]. This insight suggests that datasets trained using or compatible with Boltzmann Machines could be suitable for CRN-based modeling, providing a promising avenue for further investigation.

Another challenge lies in understanding the optimal structure of a CRN. If we can characterize the best possible structure, it would represent a significant advancement. Some well-known CRN structures, such as Detailed Balanced [10] and Complex Balanced CRNs [11], exhibit certain properties and closed-form expressions for their stationary distributions. These structures could serve as a foundation for developing more efficient models. However, for large state spaces, the computational cost of applying the **Markov Chain Tree Theorem**—especially when computing determinants of large matrices—can become prohibitively expensive. In such cases, finding the right CRN structure could help mitigate this computational burden, making the task more feasible.

Additionally, we are uncertain about the efficiency of sampling via clamping. *Theorem 1* from [8] proves that for Detailed Balanced CRNs, clamping preserves reachability, and the stationary distribution of the clamped CRN is equivalent to that of the original CRN. This result provides some confidence in the use of clamping, but further research is needed to determine its robustness and applicability in a broader range of models.

The codes used for the computations in this chapter are available in Chapter 4 in [15].

Chapter 6: Structures in CRN and Future Works

6.1 Challenges with Determinant in the Learning Rule

The learning rule derived in [\[7\]](#) incorporates the computation of the derivative of a determinant, which poses significant challenges as the state space of the energy-based models grows exponentially, typically on the order of 2^n . In such cases, the transition matrix becomes exceedingly large, with dimensions of $Q^{wh,wh}$ scaling as $(2^n - 1) \times (2^n - 1)$. Consequently, computing the derivative of determinant entails a cubic time complexity of $\mathcal{O}(8^n)$, rendering the learning process computationally intractable for large-scale models. This computational bottleneck underscores a critical limitation of the learning rule, even after leveraging techniques such as **Jacobi’s formula** to simplify the determinant’s derivative and formalize the learning rule as in Equation [\(9\)](#). Despite these efforts, the inherent expensiveness of the determinant computation remains. As The root of this computational challenge lies in the way the stationary distribution is defined—specifically, through the Markov Chain Tree Theorem (MCTT) and the Laplacian of the transition matrix. These mathematical structures, while elegant, demand significant computational resources for high-dimensional systems, thus limiting the scalability of the learning framework.

To address this challenge, we can explore two potential strategies. The first involves seeking sparse representations of the transition matrix, which could significantly reduce the computational burden of determinant evaluation. Sparse matrices often arise naturally in CRNs with structured or constrained reaction networks, where only a subset of state transitions is permitted. The second strategy involves redefining the stationary distribution by imposing a structural framework on the CRN itself. By leveraging specific properties of the CRN, such as symmetry, balance, or sparsity, we may develop alternative formulations for stationary distributions that circumvent the need for large determinant computations.

These approaches highlight the importance of structure in CRNs, not only to reduce computational costs but also to enhance the interpretability and scalability of the models. Further research into these directions could pave the way for more efficient learning algorithms in high-dimensional CRNs.

6.2 Sparse Representations

Sparse representations aim to reduce the number of non-zero entries in the transition matrix, thereby decreasing the computational overhead. For example, in Taylor Chemical Boltzmann Machines (TCBM), each state can transition to at most n other states (where n is the number of species). This significantly reduces the size of the transition matrix and brings its complexity from $\mathcal{O}(8^n)$ to $\mathcal{O}(4^n)$, which, while still exponential, is an improvement.

6.2.1 Structural Patterns in Sparse Matrices

Sparse transition matrices often exhibit specific structural patterns that further reduce complexity. In Figure 15, the sparse matrix resembles a tridiagonal-like structure. Enumerating the rows and

columns using a standard decimal representation of states highlights these patterns. There could also exist alternative enumeration schemes that transform the matrix into a 2×2 block matrix, where only the diagonal blocks are non-zero. This structure could potentially reduce the time complexity to $\mathcal{O}(2^n)$, leveraging efficient algorithms for block matrices.



Figure 15: Sparse representation of $Q^{wh,wh}$ for $|\Omega| = 2^{14}$

However, even with these optimizations, the time complexity remains prohibitive. We need to explore alternative approaches to handle the computational challenge.

6.3 Redefining Stationary Distributions Through Complex Balanced CRNs

Another approach to addressing the computational challenges in the learning rule (Equation 9) is to redefine the stationary distribution by embedding a structural framework into the CRN. Specifically, we focus on the **complex balanced condition**, a property that simplifies the form of the stationary distribution and eliminates dependence on the transition matrix.

6.3.1 Complex Balanced Condition

A CRN is called **complex balanced** if, at steady state, the net flux of reactions for every complex is zero. This condition ensures a particular equilibrium in the reaction network that simplifies the computation of the stationary distribution.

6.3.2 Stationary Distribution of a Complex Balanced CRN

For a state $x = (x_1, x_2, \dots, x_n)$, the stationary distribution in a complex balanced CRN is given by:

$$\pi(x) \propto \prod_{i=1}^n \frac{c_i^{x_i}}{x_i!},$$

where c_i are constants related to reaction parameters and species concentrations. This compact form allows the direct computation of the probability of a state based on its configuration, avoiding the need for determinant calculations or explicit construction of the transition matrix.

6.4 Structural Approaches to Enforce Complex Balanced Condition

To ensure that a CRN satisfies the complex balanced condition, we can look for two potential strategies:

1. **Imposing Structure on Parameters:** Design constraints on the rate parameters of a CRN such that the complex balanced condition is always satisfied. This involves careful calibration of reaction rates to maintain equilibrium across all complexes.
2. **Specific Structural Design:** Construct a CRN structure that inherently satisfies the complex balanced condition, regardless of the parameter values. Examples include networks derived from detailed balanced systems or specific reaction motifs known to exhibit complex balance.

6.4.1 Structure on Parameters

To express the product-form stationary distribution for a chemical reaction network (CRN), the reaction rates must satisfy the condition of being complex-balanced. The equilibrium state c emerges as a strictly positive solution derived from the mass-action kinetics equations, which define a toric dynamical system. As established in [14], this equilibrium condition corresponds to a steady state within a hypersurface, termed the moduli space, which parametrizes all rate constants ensuring the system remains toric. Moreover, *Theorem 9* in [14] highlights that this moduli space itself is a toric variety.

The challenge arises during gradient descent optimization. While updating rate parameters κ , there is no guarantee that the updated parameters will remain within this moduli space, potentially violating the complex-balanced condition. To address this, a projection-based approach must be employed: after each gradient update, the new parameters are projected back onto the moduli space of toric systems. This ensures the preservation of the complex-balanced condition, enabling the CRN to maintain the desired product-form stationary distribution while iterating toward optimal parameters.

6.4.2 Specific Structure in CRN: 0-deficiency CRN

The **deficiency** δ of a CRN quantifies its structural complexity and is defined as:

$$\delta = |\mathcal{C}| - l - s,$$

where:

- $|\mathcal{C}|$ represents the number of complexes in the CRN,
- l denotes the number of linkage classes, which are the maximal sets of complexes connected via reactions, forming single connected components in the reaction graph,
- s is the rank of the stoichiometric subspace, which characterizes all possible directions of change in species concentrations due to the network's reactions.

CRNs with a deficiency of $\delta = 0$ offer significant computational advantages. For such networks, the stationary distribution π has a closed-form expression that can be derived using combinatorial tools, as shown in the work [11].

6.5 TCBM Structure and Deficiency

The Taylor Chemical Boltzmann Machine (TCBM) structure, discussed earlier, represents a CRN with a deficiency of $\delta = 3$. While this structure is computationally tractable under certain assumptions, reducing the deficiency to $\delta = 0$ could resolve many of the challenges related to time complexity. Constructing a CRN as a zero-deficiency network inherently imposes constraints on the network's structure, simplifying computations while ensuring the stationary distribution remains product-form.


6.5.1 Advantages of Zero-Deficiency CRNs

By designing CRNs as zero-deficiency networks:

- High time complexities associated with large state spaces can be avoided,
- The network retains mathematical and structural properties conducive to analysis.

In summary, constructing CRNs with specific structures, such as zero-deficiency networks, could provide an elegant solution to computational challenges, enabling scalable and efficient modeling of chemical reaction systems.

References

- [1] *Learning System Parameters from Turing Patterns*, David Schnörr, Christoph Schnörr <https://arxiv.org/abs/2108.08542>
- [2] *Unraveling biochemical spatial patterns: Machine learning approaches to the inverse problem of stationary Turing patterns*, Antonio Matas-Gil, Robert G. Endres <https://www.sciencedirect.com/science/article/pii/S2589004224010447>
- [3] Adam Kruckman, "Markov Chain Tree Theorem", 2019 <https://akruckman.faculty.wesleyan.edu/files/2019/07/MCTT.pdf>
- [4] David F. Anderson, Gheorghe Craciun, Thomas G. Kurtz, "Product-form stationary distributions for deficiency zero chemical reaction networks". <https://arxiv.org/abs/0803.3042>
- [5] Ishaan Lamba, David Sprunger, Matthew Niemerg, "A Differentiable Programming System to Automatically Synthesize Chemical Reaction Networks from High-Level Specifications" <https://arxiv.org/abs/2302.02714>
- [6] *Automatic Differentiation: A Guide*, <https://arxiv.org/pdf/1502.05767>
- [7] *Chemical Reaction Networks: A Mathematical Approach*, <https://arxiv.org/pdf/0803.3042>
- [8] William Poole, Andrés Ortiz-Muñoz, Abhishek Behera, "Chemical Boltzmann Machines" <https://arxiv.org/abs/1707.06221>
- [9] David H. Ackley, Geoffrey E. Hinton, "A learning algorithm for Boltzmann Machine" <https://www.cs.toronto.edu/~fritz/absps/cogscibm.pdf>
- [10] William Poole, Tom Ouldridge, Manoj Gopalkrishnan, "Detailed Balanced Chemical Reaction Networks as Generalized Boltzmann Machines" <https://www.dna.caltech.edu/Papers/Poole-2022-CBMs-dbCRNs-arXiv.pdf>
- [11] David F. Anderson, David Schnoerr, Chaojie Yuan, "Time-dependent product-form Poisson distributions for reaction networks with higher order complexes" <https://people.math.wisc.edu/~dfanderson/papers/ASY2019.pdf>
- [12] Yann LeCun, Sumit Chopra, Raia Hadsell, "A Tutorial on Energy-Based Learning" <https://yann.lecun.com/exdb/publis/pdf/lecun-06.pdf>
- [13] Akaike, H. (1973). *Information theory as an extension of the maximum likelihood principle*. Pages 267-281 B. N. Petrov, and F. Csaki, (Eds.) in *Second International Symposium on Information Theory*. Akademiai Kiado, Budapest <https://gwern.net/doc/statistics/decision/1998-akaike.pdf>
- [14] *Toric dynamical systems*, Gheorghe Craciun, Alicia Dickenstein, Anne Shiue, Bernd Sturmfels <https://arxiv.org/abs/0708.3431>
- [15] To access the codes involved in this thesis, please email me at vishwas.ranjan@cbs.ac.in. The GitLab repository details are as follows: @ranjan (User ID: 492), project *MCTT_AutoDiff_CRN* (Project ID: 2471) 

¹For access to the GitLab repository, visit https://gitlab.com/ranjan/MCTT_AutoDiff_CRN