الجمهورية الجزائرية الديموقراطية الشعبية

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعليم العالي والبحث العلمي

Ministry of Higher Education and Research

UNIVERSITY OF SAIDA - Dr. MOULAY TAHAR

Faculty of Science and Technology

Department of Materials Science

THESIS

Designed for a Master's degree in Physics

Specialization: Materials Physics

Titled:

Monte Carlo simulation for Markov chain

Presented by: hadj kaddour Nasrallah

Supported on 16/06/2025 before the jury composed of:

Dr kadda Amara President	University of Saïda - Dr. Moulay Tahar
Dr Abd El kader Djaafri Supervisor	University of Saïda - Dr. Moulay Tahar
Dr Toufik Sahabi Examiner	University of Saïda - Dr. Moulay Tahar

Academic Year 2024/2025

بسم الله الرحمن الرحيم

الحمد لله الذي علم بالقلم، علم الإنسان ما لم يعلم، والصلاة والسلام على سيدنا محمد، خير معلم للبشرية، وعلى آله وصحبه أجمعين.

إن طلب العلم فريضة، وبه تسمو الأمم وترتقي، وقد شرف الله تعالى العلماء ورفع من شأنهم، فقال في محكم تنزيله:

"يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ ۚ وَاللَّهُ بِمَا تَعْمَلُونَ خَبِيرٌ"

(سورة المجادلة، الآية 11)

وإيمانًا منا بأن التوفيق أولاً وأخيرًا من الله عز وجل، فقد كان دعاؤنا الدائم:

"رَبِّ زِدْنِي عِلْمًا"

(سورة طه، الآية 114)

وفي ختام هذا العمل المتواضع، لا يسعنا إلا أن نحمد الله ونشكره على ما أنعم به علينا من توفيق، ونقول كما قال سليمان عليه السلام:

"...رَبِّ أَوْزِعْنِي أَنْ أَشْكُرَ نِعْمَتَكَ الَّتِي أَنْعَمْتَ عَلَيَّ"

(سورة النمل، الآية 19)

نسأل الله أن يكون هذا الجهد خالصًا لوجهه الكريم، وأن ينفع به، ويجعله لبنة في بناء مستقبل علمي مشرق

thanks

I extend my sincere thanks and gratitude to everyone who contributed to the success of this thesis, especially my supervisor [Sahabi Tawfik] for his valuable support and guidance throughout the process. I also cannot fail to thank all the professors who provided me with knowledge and wisdom throughout my university career.

I must also express my gratitude to my esteemed family, who have been a constant source of support and moral encouragement, and to everyone who has stood by me and offered me encouragement.

My heartfelt thanks to everyone who contributed, even with a kind word.

Dedication

To those who have had the greatest hand in bringing me to this moment...

To my dear parents, my support and the light of my path, who instilled in me a love of learning and were my greatest support.

To my esteemed teachers, who spared no effort in offering their knowledge and guidance, serving as a beacon that illuminated my path.

To my friends and colleagues who shared my journey of struggle and success and left their mark on my heart before my memoir.

And to everyone who believed in me, encouraged me, and pushed me forward...

I dedicate this humble work in gratitude and appreciation

Table of Contents

List of Figures

Liste of tables

Abstract

This work addresses an inverse problem in channel biophysics: reconstructing the transition matrix of a Markov model governing ion channel states from observed trajectories. Using a Metropolis Monte Carlo (MCMC) approach, we estimate the matrix by minimizing the mismatch between predicted and observed state transitions. Simulations reveal that temperature in MCMC critically affects convergence and accuracy. Among several tested values, a moderate temperature (T = 1.0) achieved the closest match to the true matrix, demonstrating that careful temperature control is essential for accurately capturing ion channel dynamics from data.

Résume

Ce travail traite un problème inverse en biophysique des canaux ioniques : la reconstruction de la matrice de transition d'un modèle de Markov à partir de trajectoires observées. En utilisant une méthode de Monte Carlo par algorithme de Metropolis (MCMC), nous estimons la matrice en minimisant l'écart entre les transitions d'état prédites et observées. Les simulations montrent que la température dans l'algorithme MCMC influence fortement la convergence et la précision. Parmi les différentes températures testées, une température modérée (T = 1.0) fournit la meilleure estimation de la matrice réelle, soulignant l'importance du contrôle de la température pour modéliser avec précision la dynamique des canaux ioniques à partir des données.

ملخص

يتناول هذا العمل مسألة عكسية في الفيزياء الحيوية لقنوات الأيونات، تتمثل في إعادة بناء مصفوفة الانتقال لنموذج ماركوف انطلاقاً من مسارات الحالات المرصودة. استخدمنا طريقة مونتي كارلو عبر خوارزمية متروبوليس (MCMC) لتقدير المصفوفة من خلال تقليل الفرق بين الانتقالات المتوقعة والمرصودة. أظهرت المحاكاة أن درجة الحرارة في خوارزمية MCMC تؤثر بشكل كبير على سرعة التقارب ودقة النتائج. ومن بين القيم المختبرة، حققت درجة الحرارة المتوسطة ($T = 1.0$) أفضل تطابق مع المصفوفة الحقيقية، مما يبرز أهمية التحكم بدرجة الحرارة لتمثيل ديناميكية قنوات الأيون بدقة انطلاقاً من البيانات

General introduction

General introduction

Markov chains are a class of stochastic models that describe systems transitioning between a finite or countable number of states according to probabilistic rules. First introduced by Andrey Markov in the early 20th century, they are characterized by the Markov property, which asserts that the future state of the system depends only on the present state and not on the sequence of events that preceded it [1]. Mathematically, a discrete-time Markov chain (DTMC) is defined by a set of states S = $\{s_1 s_2 \dots s_n\}$ and a transition matrix P where each element p_{ij}represents the probability of transitioning from state sis_i to state sjs_j in one time step. These models have become foundational in various domains such as physics, biology, finance, and artificial intelligence [2].

The power of Markov chains lies not only in their simplicity but also in their ability to model equilibrium behavior through long-run distributions. The stationary distribution of a Markov chain describes the probability of finding the system in each state after a large number of transitions, assuming the chain is irreducible and aperiodic. In statistical physics, for instance, Markov chains underpin models such as the Ising model for ferromagnetism and are central to approaches in computational biology for modeling molecular evolution and protein folding [3].

Monte Carlo Simulation

Monte Carlo (MC) simulations are a class of computational algorithms that rely on repeated random sampling to obtain numerical results. When combined with Markov chains, they form Markov Chain Monte Carlo (MCMC) methods, which are widely used for sampling from complex probability distributions. The most popular MCMC algorithms include the Metropolis-Hastings algorithm and the Gibbs sampler, both of which construct a Markov chain whose stationary distribution corresponds to the target distribution of interest [4].

These simulation techniques are indispensable in fields where analytical solutions are infeasible due to high dimensionality or complex interactions. For example, in Bayesian statistics, MCMC allows for estimation of posterior distributions that are otherwise computationally intractable [5]. Similarly, in materials science and statistical mechanics, Monte Carlo methods provide estimates of thermodynamic quantities and help simulate systems near equilibrium [6].

While traditional applications of Markov chains and MCMC involve forward modeling—using known transition probabilities to simulate behavior—there is growing interest in the inverse Markov chain problem. This involves deducing the transition matrix or the generator of a Markov process from observed data, such as empirical state transitions or stationary distributions [7].

The inverse problem is of great importance in data-driven modeling, particularly when the underlying stochastic dynamics are not directly observable. Applications range from inferring hidden transition networks in biological systems to estimating the movement patterns in social or economic systems [8]. Several approaches have been developed, including maximum likelihood estimation, Bayesian inference, and regularized optimization techniques, to reconstruct the most likely Markov model given partial or noisy data [9].

Recent advances have extended inverse problems to non-equilibrium systems and coarse-grained representations, enabling more accurate reconstructions even in the presence of temporal correlations or sampling limitations. For example, in molecular dynamics, inverse methods are used to determine effective transition rates between conformational states, enabling reduced models for large biomolecular systems [10]

References

[1] A. A. Markov, "An Example of Statistical Investigation of the Text Eugene Onegin Concerning the Connection of Samples in Chains," *Science in Context*, vol. 19, no. 4, pp. 591–600, 1906.

[2] Norris, J. R. *Markov Chainse*. Cambridge University Press, 1997.

[3] Levin, D. A., Peres, Y., & Wilmer, E. L. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.[3]

[4] Metropolis, N., et al., "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[5] Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1996.

[6] Landau, D. P., & Binder, K. *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, 2000.

[7] Ghosh, S. K., & Varadhan, S. R. S., "Inverse Problems for Markov Chains," *Statistical Science*, vol. 6, no. 4, pp. 432–448, 1991.

[8] Pan, W., et al., "Inferring Markov Chain Transition Models from Time Series Data," *Bioinformatics*, vol. 21, no. 24, pp. 4350–4356, 2005.

[9] Wu, H., Noé, F., & Wehmeyer, C., "Variational Koopman Models: Inferring Dynamics from Observations," *Journal of Chemical Physics*, vol. 153, 2020.

[10] Bowman, G. R., Pande, V. S., & Noé, F. *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation*. Springer, 2014.

Chapter 1

Generalities about the Markov chain

1.1Introduction

A Markov chain is a type of stochastic process used to model systems that evolve over time where the outcome at any given time depends only on the current state — not the sequence of events that preceded it. This characteristic is called the Markov property, or memory lessness.

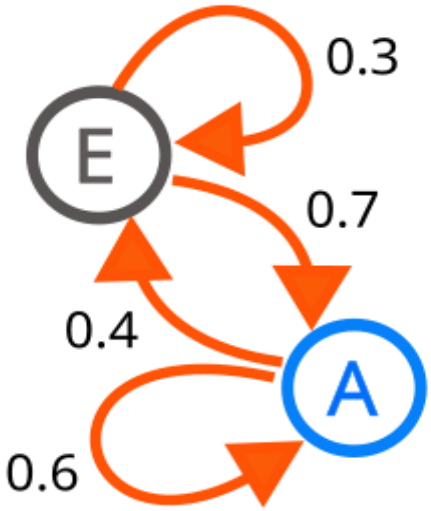

Figure (1.1) *Representing a two-state Markov process. The numbers are the probability of changing from one state to another stat*

Markov chains provide a mathematical framework to (Fig. 1.1) study random processes where: Time is discrete (i.e., steps happen one at a time). The system is in one of a finite (or countable) number of states, and the transition from one state to another is governed by a set of fixed probabilities. Transition Probabilities: Probabilities of moving from one state to another, denoted as P (i, j), the probability of moving from state i to state j. The transition Matrix (P) is a matrix representing all the transition probabilities between states. Initial Distribution: The starting probability of being in each state

1.2 Definition

A Markov chain is a mathematical stochastic model that describes a sequence of possible events or states in which the outcome of each event depends only on the current state (positions that the system can be in), not on the events that preceded it [1]. This property is called the Markov property, or the "memory less" property. The Transition Probabilities are the probabilities of moving from one state to another. These probabilities are typically represented in a matrix form, called the transition matrix [2]. We distinguish two types of Markov Chains: *Discrete-Time Markov Chain*: The system transitions from one state to another are in discrete time steps.[3] *Continuous-Time Markov Chain*: The system transition can be at any point in time.[4]

1.3 History

Markov chains are a fundamental concept in mathematics, statistics, and probability theory. Their history dates back to work in the field of probability and linear algebra in the early 20th century. Before Markov chains, concepts related to conditional probabilities and the evolution of stochastic systems were already under discussion. The name "Markov chain" comes from the Russian mathematician Andrey Markov, who developed these ideas in the early 20th century. In 1906, Markov introduced what is now known as Markov chains in his paper "*On the chain of probabilities in phenomena that follow a law of dependence*". He studied stochastic processes where the future depends only on the current state, and not on past states

This concept revolutionized probability by allowing complex systems to be modeled in a simpler and more understandable way. Markov also developed the notion of Markov property, which means that the probability of transition from one state to another depends only on the current state (and not on how one arrived at that state). Markov's work has been refined over the decades by other mathematicians. For example, Kolmogorov made

major contributions in the 1930s to formalize Markov chains using modern concepts of probability and state space.

In the 1940s and 1950s, Markov chains began to be used in fields such as queuing theory, dynamical systems analysis, and statistical physics. Markov chains are a fundamental concept in mathematics, statistics, and probability theory. Their history dates back to work in the field of probability and linear algebra in the early 20th century.Before Markov chains, concepts related to conditional probabilities and the evolution of stochastic systems were already under discussion. The name "Markov chain" comes from the Russian mathematician Andrey Markov, who developed these ideas in the early 20th century. In 1906, Markov introduced what is now known as Markov chains in his paper "*On the chain of probabilities in phenomena that follow a law of dependence*". He studied stochastic processes where the future depends only on the current state, and not on past states. This concept revolutionized probability by allowing complex systems to be modeled in a simpler and more understandable way. Markov also developed the notion of Markov property, which means that the probability of transition from one state to another depends only on the current state (and not on how one arrived at that state). Markov's work has been refined over the decades by other mathematicians. For example, Kolmogorov made major contributions in the 1930s to formalize Markov chains using modern concepts of probability and state space.In the 1940s and 1950s, Markov chains began to be used in fields such as queuing theory, dynamical systems analysis, and statistical physics. *"Zur Theoria der Markoff Chen Ketten.* «German publication that extended the theory, especially in continuous-time Markov processes.[5]

1.4 Example of a Simple Markov Chain

Consider a weather system (Table 1.1) with two possible states: ***Sunny (S),*** and ***Rainy (R).*** The transition probabilities might look like this

From/to	Sunny(s)	Rainy(R)
Sunny(s)	0.8	0.2
Rainy(R)	0.4	0.6

Table 1.1 Example of transition matrix

The associated transition matrix is then

$$T = \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix} \qquad (1.1)$$

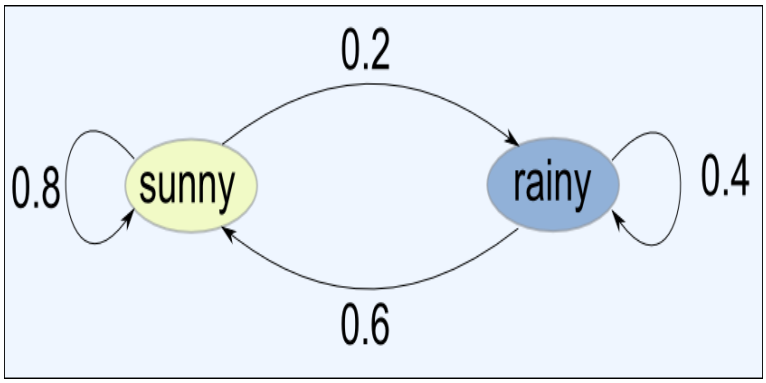

Figure (1.2) *Graph of transition*

We Start at an initial state (e.g., Sunny). Then, we use the transition probabilities to determine the next state (e.g., if Sunny, Fig. (1.2) the next state could be Sunny with probability 0.8 or Rainy with probability 0.2). We repeat these two steps to evolve the system over time.

1.5 Applications of Markov Chains

1.5.1 Physics

Markovian systems appear extensively in thermodynamics and statistical mechanics, whenever probabilities are used to represent unknown or unmodelled details of the system, if it can be (Fig. (1.3)) assumed that the dynamics are time-invariant, and that no relevant history need be considered which is not already included in the state description. For example, a thermodynamic state operates under a probability distribution that is difficult or expensive to acquire. Therefore, Markov Chain Monte Carlo method can be used to draw samples randomly from a black-box to approximate the probability distribution of attributes over a range of objects.

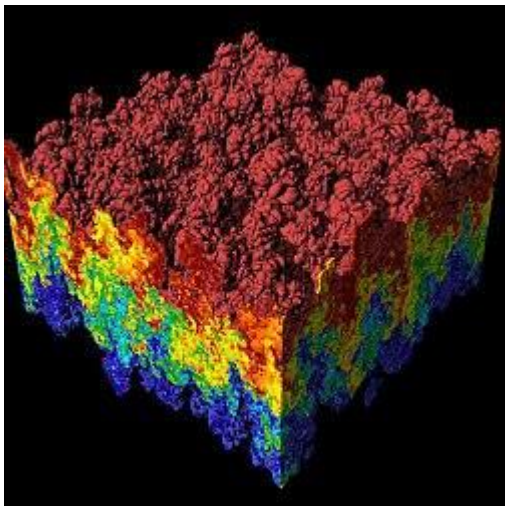

Figure (1.3) Mechanics electromagnetics Multiphysics Particle physics Thermodynamics Simulation

1.5.2 Weather Prediction

Markov Chains model weather patterns by representing different weather states (e.g., sunny, rainy) and the probabilities of transitioning between them. Let **P** be the transition matrix, then we have

$$P = \begin{bmatrix} p_{sunny \to sunny} & p_{sunny \to rainy} \\ p_{rainy \to sunny} & p_{rainy \to rainy} \end{bmatrix} \quad (1.2)$$

1.5.3 Google PageRank

PageRank uses a Markov Chain to rank web pages based on their link structure.

$$\pi = \pi p \quad (1.3)$$

1.5.4 Text Generation (Markov Chain Text Models)

Markov Chains generate text by modeling the probability of a word following another, based on observed frequencies.

P ($\omega_{n+1}|\omega_n$)=probability of next word given current word P(ω_n)word

1.5.5 Solar irradiance variability

Inventory and Queueing Systems

X_{n+1}=f (X_n), demand with probabilistic demand"

1.5.6 Speech recognition

Hidden Markov models have been used in automatic speech recognition systems.[6]

1.5.7 Queueing theory

Markov chains are the basis for the analytical treatment of queues (queueing theory). Agner Krarup Erlang initiated the subject in 1917.[1] This makes them critical for optimizing the performance of telecommunications networks, where messages must often compete for limited resources (such as bandwidth).[8] Numerous queueing models use continuous-time Markov chains. For example, an M/M/1 queue is a CTMC on the non-negative integers where upward transitions from i to $i + 1$ occur at rate λ according to a Poisson process and describe job arrivals, while

transitions from i to $i-1$ (for $I>1$) occur at rate μ (job service times are exponentially distributed) and describe completed services (departures) from the queue.

1.6 Probability properties

Discrete-time Markov chain is a sequence of random variables X_1, X_2, X_3, … with the Markov property, namely that the probability of moving to the next state depends only on the present state and not on the previous states

$$P(X_{n+1}=x|X_n=x_n, X_{n-1}=x_{n-1},\dots,X_0=x_0) = P(X_{n+1}=x|X_n=x_n) \quad (1.4)$$

probabilities are well defined, that is

$$P(X_1=x_1, X_2=x_2,\dots,X_n=x_n)>0 \quad (1.5)$$

The possible values of X_i form a countable set S called the state space of the chain. Time-homogeneous Markov chains are processes were

$$(X_{n+1}=x|, X_n=y) = P(X_n=x|, X_{n-1}=y) \quad (1.6))$$

for all n. The probability of the transition is independent of n. Stationary Markov chains are processes were

$$P(X_0=x_0, X_1=x_1,\dots,X_k=x_k) = P(X_n=x_0, X_{n+1}=x_1,\dots,X_{n+k}=x_k) \quad (1.7)$$

for all n and k. Every stationary chain can be proved to be time-homogeneous by Bayes' rule. A necessary and sufficient condition for a time-homogeneous Markov chain to be stationary is that the distribution of X_0 is a stationary distribution of the Markov chain. A Markov chain with memory (or a Markov chain of order m) where m is finite, is a process satisfying

$$P(X_{n+1}=x|X_n=x_n, X_{n-1}=x_{n-1},\dots,X_0=x_0) = P(X_{n+1}=x|X_n=x_n, X_{n-1}=x_{n-1},\dots,X_{n-m}=x_{n-m}) = \quad (1.8)$$

In other words, the future state depends on the past m states. It is possible to construct a chain (Y_n) from (X_n) which has the 'classical' Markov property by taking as state space the ordered *m*-tuples of *X* values, i.e.

$$Y_n = (X_n, X_{n-1}, \dots X_{n-m+1}) \tag{1.9}$$

1.7 Other properties

1.7.1 Stationary Distribution

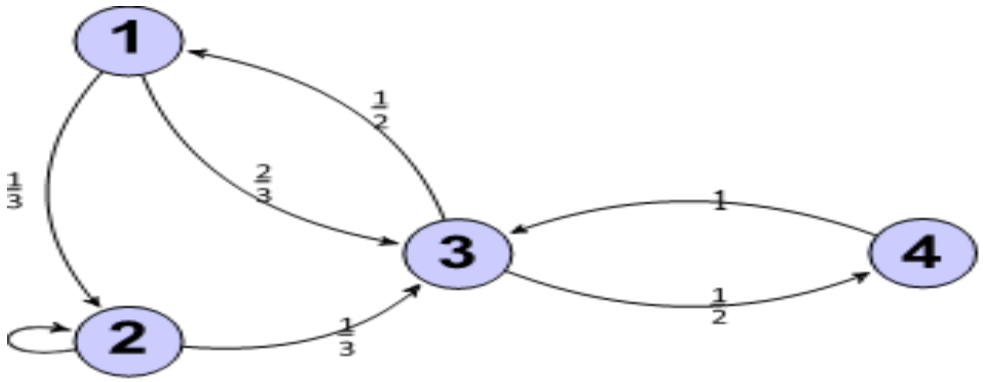

Figure 1.5 : Periodicity is an irreducible class property.

A stationary distribution π is a probability (fig (1.5)) distribution over the states that remain unchanged after transitions. The stationary distribution satisfies

$$\pi T = \pi \tag{1.10}$$

Where π is the row vector representing the stationary distribution, and *T* is the transition matrix. This ensures that the probability distribution π is invariant under the transition matrix.

1.7.2 Irreducibility

A Markov Chain is irreducible if it is possible to reach any state from any other state. For irreducibility, for any pair of states (*i, j*), there must exist some $n > 0$ such that

$$P^n(i,j) > 0 \qquad (1.11)$$

This means that there is a non-zero probability of transitioning from state *i* to state *j* in a finite number of steps.

1.7.3 Ergodicity

A Markov Chain is ergodic if it is irreducible and a periodic, meaning the chain will eventually reach the stationary distribution, regardless of the initial state. If a chain is ergodic, it satisfies the condition that, for any state *i*, there is a nonzero probability of reaching any state *j* in a finite number of steps, and that state *i* will eventually return with probability

$$P^n(i;j) > 0 \quad for\ some\ n \geq 1 \qquad (1.12)$$

where $P^n(i;j)$ is the probability of reaching state j from state iii in N steps. The ergodicity condition guarantees that the chain will eventually converge to a stationary distribution.

1.7.4 Reversibility

Description: A Markov Chain is reversible if it satisfies the detailed balance equations, meaning that the rate of transition from state iii to state j is the same as from state j to state iii, weighted by the stationary distribution.

$$\pi_i P_{ij} = \pi_j P_{ij} \qquad \forall i, j \qquad (1.13)$$

This equation ensures that the process is reversible and in equilibrium when the chain reaches the stationary distribution π

1.7.5 Absorbing States

Description: A state is absorbing if, once entered, the system cannot leave it. An absorbing Markov Chain contains at least one absorbing state.

Let *P* be the transition matrix. If a state iii is absorbing, the probability of transitioning to another state from iii is zero:

$$P_{ij}=1;\ P_{ij}=0 \quad \text{for I} \neq j \tag{1.14}$$

If Q is the submatrix of the transition matrix corresponding to transient states and RRR is the matrix of transitions from transient to absorbing states, the fundamental matrix NNN is given by

$$N=(I-Q)^{-1} \tag{1.5}$$

This matrix helps calculate the expected number of visits to each transient state before absorption.

1.7.6 Transience and Recurrence

Description: States can be either recurrent (eventually revisited) or transient (never revisited). For a state to be recurrent, the expected number of steps to return to state iii should be finite:

$$E_i = \sum_{n=1}^{\infty} \mathrm{n.P(Xn = i} | x_0 = \mathrm{i}) \tag{1.16}$$

If finite, state i is recurrent. If E_i is infinite, state iii is transient.

1.7.7 Periodic States

Description: A state iii is periodic if there exists an integer d>1 such that If the period of a state iii is did, it satisfies:

$$D=gad\ \{n:P^n\ (i;\ i) >0\} \tag{1.17}$$

A Markov Chain is aperiodic if all states have period 1, meaning the process can return to any state at irregular time intervals.

1.7.8 Convergence to Stationarity

Description: For an ergodic Markov Chain, the chain will converge to the stationary distribution regardless of the starting distribution.

(Convergence): The convergence of the chain to its stationary distribution can be measured by the total variation distance between the current distribution and the stationary distribution π

$$\mathrm{dtv}(\mathrm{u_n}, \pi) = 1/2 \sum_i |\mathrm{p^n}(i) - \pi(i)| \tag{1.18}$$

this distance approaches zero if the chain is ergodic.

1.7.9 Mixing Time

Description: The mixing time of a Markov Chain is the time it takes for the chain to come close to its stationary distribution. The mixing time is the smallest time such that the total variation distance between the distribution and the stationary distribution π\piπ is less than ϵ\epsilonϵ:

$$d_{T_v}(p^t, \pi) < \in \tag{1.19}$$

This gives an upper bound on how quickly the chain "forgets" its initial state and reaches the stationary distribution.

1.8 Conclusions

In a Markov chain, future states depend only on the current state, and not on the sequence of events that preceded it. This simplifies modeling and analysis but also limits its application in systems where the history is important. The system evolves based on a set of transition probabilities, which are typically represented in a matrix form (transition matrix). The probability of moving from one state to another can be calculated using these transition matrices. While Markov chains are powerful, they may not be suitable for modeling systems where past states or more complex dependencies influence future behavior. Also, for certain chains,

convergence to a steady state may not always occur or could take an impractically long time.

References

[1] A. A. Markov (1906) "Rasprostranenie zakona bol'shih chisel na velocity, zavisyaschie drug to druga". Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom Universiteit, 2-ya seriya, volume 15, pp.

[2] A. A. Markov (1971). Extension of the limit theorems of probability theory to a sum of variables connected in a chain". reprinted in Appendix B of: R. Howard. *Dynamic Probabilistic Systems, volume 1: Markov Chains*. John Wiley and Sons [3] Classical Text in Translation: Markov, A. A. (2006). An Example of Statistical Investigation of the Text Eugene Onegin Concerning the Connection of Samples in Chains". Science in Context. 19 (4) Translated by Link, David: 591–600

[4] S. P. Meyn and R. L. Tweedie (1993) *Markov Chains and Stochastic Stability*. London: Springer-Verlag ISBN 0-387-19832-6. online: MCSS . Second edition to appear, Cambridge University Press, 2009

[5] S. P. Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, 2007. ISBN 978-0-521-88441-9. Appendix contains abridged Meyn & Tweedie. online: CTCN

[6] "Student Chapters". Society for Industrial and Applied Mathematics. Retrieved 30 August 2017 Markov Chains" by J. R. Norris (1998) – This paper provides an in-depth explanation of the theory behind Markov chains and their mathematical properties.

[7] Markov Chains" by J. R. Norris (1998) – This paper provides an in-depth explanation of the theory behind Markov chains and their mathematical properties.

Chapter 2

Monte Carlo simulation

2.1 Introduction

This Monte-Carlo method, or the Monte-Carlo method, is an algorithmic method that is used to calculate a numeric value applied in the use of advanced algorithms, it is the easiest of the probabilistic techniques.

Figure (2.1) :Monte Carlo Simulation: What It Is, How It Works, History, 4 Key Step

The Monte-Carlo methods are participatory utilizes to (fig 2.1) calculate integrals of larger dimensions than 1 (in particular, to calculate surfaces and volumes). These are suitable for use in specific physics, or simulations cause problems that allow for the signal format or the sensitivity of the detector. Comparison of the best simulations in these simulations may allow you to provide information on specific characteristics, including new parts. The Monte-Carlo simulation method allows us to introduce a statistical estimate of the risk in a precise financial decision. It consists of isolating the variables in the projector, telling you the difference or the margin, and it affects one of the possibilities. Because of these factors, a large number of household tires, which will affect the previous pre-determined risk factors, will have an effect, in order to find out the risk of occurrence of the results. Here is the example, the selection of the collection mode in the

public private partnership (PPP) cadre can be analyzed by the Monte Carlo method, so that it can calculate the risk sharing between public and private actors. On parle de "risks valorizes" or "values à risk». The veritable development of Monte-Carlo methods has the greatest effect on the impulsion of John von Neumann and Stanislaw Ulam notation, when the Second World War, and searches on the automatic bomb fabrication. It also uses specific methods to solve the needs of parts in the Monte-Carlo N-Particle transport (MCNP) cadre.the name of these methods, which helps all those who have practiced at the Monte-Carlo Casino, was created in 1947 by Nicholas Metropolis1, and was originally published in 1949 in a co-written article with Stanislaw Ulam2.

2.2 Convergence and Statistical Accuracy

After a sufficient number of iterations, the results of the Monte Carlo simulation converge to a stable value, which can be considered an estimate of the true result.: this convergence helps assess the accuracy of the simulation and provides a way to quantify uncertainty in predictions.

2.2.1 Output Analysis

Once the simulations have been completed, the output (i.e., the results of each trial) is analyzed using statistical methods. The outputs may be used to calculate metrics like the mean, variance, percentiles, or probabilities of certain outcomes, helping to draw conclusions about the system's behavior or make informed decisions.

2.2.2 Sensitivity Analysis

Definition: Sensitivity analysis involves studying how sensitive the results of the simulation are to changes in the input variables. Purpose: This helps identify which variables have the most significant impact on the outcome, allowing for better risk management and prioritization of factors in a model.

2.2.3 Variance Reduction Techniques

These are methods used to reduce the number of simulations needed to achieve a certain level of accuracy. Examples include importance sampling, stratified sampling, and antithetic variates. Variance reduction techniques can make Monte Carlo simulations more efficient by lowering computational costs without sacrificing accuracy.

2.2.4 Sampling Methods

There are different ways to generate random samples for the simulation. Common sampling techniques include: Simple Random Sampling: Randomly selecting inputs for each simulation from the distribution. Stratified Sampling: Dividing the population into subgroups (or strata) and sampling within each subgroup to reduce variance. Latin Hypercube Sampling: A statistical method to ensure that all areas of the input space are sampled more evenly.

2.3 Steps in a Monte Carlo Simulation

Monte Carlo simulations are typically used to model complex systems and estimate the impact of uncertainty or variability. The process involves a series of well-defined steps to simulate and analyze the system. Here's a breakdown of the typical steps involved

2.3.1 Define the Problem or System

The objective is clearly outline what you are trying to model or analyze and identify the system's variables, inputs, and outputs, and what is the goal of the simulation. For example, estimating the probability of a specific outcome, optimizing a decision, or evaluating risk

2.3.4 Identify Input Variables and Define Probability Distributions

Identify the uncertain variables in your system and specify their probability distributions. Each input will be treated as a random variable with a known distribution (e.g., normal, uniform, or exponential).Use historical data, expert opinions, or assumptions to define the distributions of each uncertain variable. If modeling stock price, you may assume the returns follow a normal distribution with a mean of 5% and a standard deviation of 10%.

2.3.5 Generate Random Samples

Objective: For each uncertain variable, generate random values according to the defined probability distributions. These values represent possible outcomes of the uncertain inputs. Random sampling can be done using tools like random number generators. For each input, you may use methods like inverse transform sampling, rejection sampling, or other techniques depending on the distribution.

2.3.6 Run the Simulation (Perform One Trial)

Objective: Use the randomly generated input values to simulate one possible outcome of the system. Calculate the output or result for that trial.

Details to consider: Apply the input values to the model and calculate the desired output, which could be a profit, cost, risk, or some other metric. If simulating an investment's future value, use the randomly generated return and time period to calculate the investment's final value.

2.4 Repeat the Simulation (Multiple Trials)

Run the simulation many times (e.g., 1,000, 10,000, or more) with different random samples each time. This helps to capture the variability and uncertainty in the system. The more trials you run, the more accurate and

reliable your results will be. Each iteration uses a new set of random samples, providing a broad range of possible outcomes.

2.4.1 Analyze the Results

After performing a large number of trials, analyze the results to make inferences or decisions. Common statistical analyses include calculating the mean, standard deviation, percentiles, and probabilities. Visualization methods such as histograms, boxplots, or cumulative distribution functions (CDFs) can also help interpret the results.

2.4.2 Make Decisions Based on Results

Use the insights gained from the simulation to make informed decisions, assess risks, or determine the best course of action.the results help in understanding the risk and uncertainty in the system, guiding decision-making under uncertainty. You may use these results to set expectations, adjust strategies, or optimize choices.

2.4.3 Validate and Refine the Model (Optional)

Validate the simulation results and refine the model if necessary. Compare the simulation results to real-world data (if available) or use expert judgment to ensure that the model's assumptions are reasonable. If discrepancies arise, you may adjust the input distributions or the model itself.

2.5 Applications of Monte Carlo Simulations

2.5.1: Monte Carlo integration

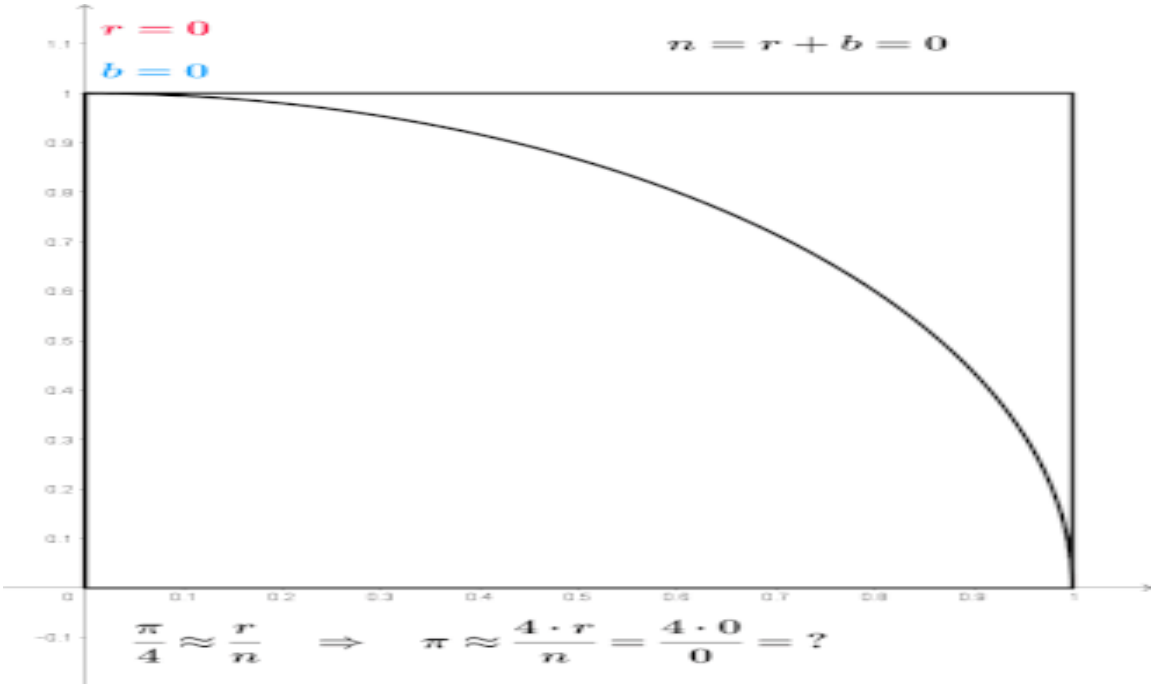

Figure (2.2): Monte Carlo method applied to approximating the value of π

A classic example is estimating the (fig (2.2)) integral of a function f(x) over an interval [a.b]

$$\int_a^b f(\mathrm{x})\,\mathrm{dx} \approx \frac{b-a}{N}\sum_{i=1}^{N} f(xi) \qquad (2.1)$$

Where :

xi are random samples uniformly drawn from the interval [a, b]:

N is the number of samples

2.5.2 estimating π using Monte Carlo

A common introductory example involves estimating π by simulating points in a unit square and checking how many fall inside a quarter circle

$$\pi \approx 4 \frac{Number\ of\ points\ inside\ quarter\ circle}{total\ number\ of\ boints} \quad (2.2)$$

If we randomly generate (xi. Yi) such that $0 \leq xi.Yi \leq 1$ then a point is inside the quarter circle if:

$$x^2 + y^2 \leq 1 \quad (2.3)$$

2.5.3 General Monte Carlo Estimation Formula

To estimate the expected value E[f(X)] for a random variable X:

$$E\ [f(x)] \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i) \quad (2.4)$$

2.5.4 Finance and Investment

Monte Carlo Simulation is used to forecast the possible returns of a financial portfolio by simulating random paths of asset prices over time, capturing market volatility and uncertainty.

2.5.5 Project Management

Uncertainty in task durations is modeled to estimate the likelihood of project completion dates using Monte Carlo simulation. A software company simulates uncertain task durations to assess the probability of delivering a product on time.

$$T = \sum_{i=1}^{n} ti \quad (2.5)$$

2.5.6 Healthcare and Medicine

Monte Carlo simulations model how radiation interacts with tissues to calculate dose distributions more accurately than deterministic methods.

Physicians use Monte Carlo-based treatment planning to adjust tumor doses in proton therapy.

$$D = \sum_{i=1}^{n} \frac{Ei}{m} \tag{2.6}$$

Verhaegen, F., & Suenens, J. (2003). "Monte Carlo modelling of external radiotherapy photon beams." *Physics in Medicine and Biology*, 48(21), R107.
https://doi.org/10.1088/0031-9155/48/21/R01 [1]

2.5.7 Environmental Science

Monte Carlo simulations are used to estimate the likelihood of extreme rainfall and flooding by generating thousands of random weather scenarios.

2.6 Conclusion

The main purpose of this chapter is to present algorithms based on various Monte Carlo methods and techniques, in addition to studying Markov chains. We have followed the chronological order of events in the evolution and introduction of the various approaches to approximating integrals, either analytically, numerically, or by simulation. The criticisms made of the first two approaches point us toward the simulation approach, whose adaptive preferential sampling methods and MCMC methods prevail over the other techniques we have presented.

2.7 References

[1] Robert, C. P., & Casella, G. (2004). Title: *Monte Carlo Statistical Methods* (2nd Edition) Publisher: Springer ISBN: 978-1-4757-4145-2
Glasser man, P. (2003). Title: *Monte Carlo Methods in Financial* [2] *Engineering* Publisher: Springer (Stochastic Modelling and Applied Probability, Vol. 53) ISBN: 978-0-387-21617-1
[3] Kroese, D. P., Taimre, T., & Botev, Z. I. (2013). Title: *Handbook of Monte Carlo Methods* Publisher: Wiley ISBN: 978-0-470-17793-8

[4] Monte Carlo Markov Chain (MCMC) methods are a class of algorithms that rely on Markov chains to perform Monte Carlo integration, widely used in Bayesian statistics, computational physics, machine learning, and statistical inference.

[5] Brooks, S., Gelman, A., Jones, G. L., & Meng, X.-L. (Eds.). (2011). Title: *Handbook of Markov Chain Monte Carlo*
[6] Publisher: CRC Press ISBN: 978-1420079418
Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Title: *Equation of State Calculations by Fast Computing Machines* Journal: *The Journal of Chemical Physics*, 21(6), 1087–1092

[7] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Title: *Bayesian Data Analysis* (3rd Edition) Publisher: CRC Press ISBN: 978-1439840955
[8]Neal, R. M. (1993). Title: *Probabilistic Inference Using Markov Chain Monte Carlo Methods* Technical Report: University of Toronto

Chapeter:3

Inverse Markov problem by monte Carlo simulation

3.1 Introduction

We present a biophysics example of a transition matrix in a Markov process, modeling ion channel gating dynamics. It is a classic application in neurobiology and electrophysiology. [1, 2] In traditional (direct) applications, the transition probability matrix is given, and the evolution of the system is simulated or analyzed from this known matrix. However, in many practical situations, we observe only the system trajectories—sequences of visited states—without direct access to the transition probabilities themselves. The challenge in such cases is to infer the transition matrix that best explains the observed dynamics. This is an inverse problem, and it is typically solved in this chapter using Metropolis alghorithm.

3.2 Ion channel

Ion channels are **specialized proteins** embedded in cell membranes that allow ions (e.g., K^+, Na^+, Ca^{2+}, Cl^-) to pass in and out of cells. They play critical roles in: **Electrical signaling** (e.g., nerve impulses, Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). *Equation of state calculations by fast computing machines*. Journal of Chemical Physics, 21(6), 1087–1092.[2] muscle contraction), **Maintaining cellular homeostasis** (e.g., pH, volume), and **Sensory processes** (e.g., hearing, taste). Channels switch between **open** (conducting) and **closed** (non-conducting) states, controlled by:

- **Voltage** (e.g., voltage-gated Na^+ channels in neurons).
- **Ligands** (e.g., neurotransmitters like GABA activating Cl^- channels).
- **Mechanical force** (e.g., hair cells in the inner ear).

3.3 Markov model

3.3.1 Continuous matrix Q

A potassium (K^+) ion channel fluctuates between three conformational states:

- **Closed (C)** – Channel is shut, no ions pass.
- **Open (O)** – Channel allows ion flow.
- **Inactivated (I)** – Channel is blocked (temporarily non-conducting).

The **transition rate matrix** Q for a potassium (K^+) ion channel describes the stochastic switching between its conformational states (e.g., Closed, Open, Inactivated). Below, we rigorously derive the elements of Q and explain their biophysical meaning. Experimental data [3] suggests the following **transition rates (per millisecond)**:

C → O: Rate = 0.3 ms^{-1}

O → C: Rate = 0.4 ms^{-1}

O → I: Rate = 0.1 ms^{-1}

I → C: Rate = 0.2 ms^{-1}

Where, no direct transitions $C \leftrightarrow I$ or $I \rightarrow O$ are allowed in this model. Then, we get the channel Q matrix

$$Q = \begin{bmatrix} -0.3 & 0.3 & 0 \\ 0.4 & -0.5 & 0.1 \\ 0.2 & 0 & -0.2 \end{bmatrix} \quad (3.1)$$

Where, r**ows sum to 0,** and diagonal entries Q_{ii} are negative (to ensures probability conservation). The **off-diagonal elements** $Q_{i \neq j}$ are the transition rates from state i to j. Then, we have:

- **Fast activation**: High $C \rightarrow O$ rate ($Q_{CO} = 0.3$) ensures quick response to depolarization.
- **Slow inactivation**: Low $O \rightarrow I$ rate ($Q_{OI} = 0.1$) prevents premature channel blockage.
- **Recovery**: $I \rightarrow C$ rate ($Q_{IC} = 0.2$) determines refractory period.

3.3.2 Discrete matrix P

For a **small-time step** Δt (e.g., 0.1 ms), the transition probability matrix P is approximated as

$$P \approx I + Q\Delta t \tag{3.2}$$

where I is the identity matrix. For $\Delta t = 0.1\,ms$, we get

$$P = \begin{bmatrix} 0.97 & 0.03 & 0 \\ 0.04 & 0.95 & 0.01 \\ 0.02 & 0 & 0.98 \end{bmatrix} \tag{3.3}$$

We note that all rows sums to 1. The probability of staying in **Closed (C)** for 0.1 ms is **97%, the p**robability of the transition **C → O** in 0.1 ms is **3%**. Note also that $P_{ii} \approx 1$ implies slow dynamics.

3.3.3 Steady-State Probabilities

The steady state π is given by one of the two following equations

$$\pi . Q = 0 \text{ or } \pi . P = \pi \tag{3.4}$$

If $\pi = [\pi_C, \pi_O, \pi_I]$, then we get

$$[\pi_C, \pi_O, \pi_I] \begin{bmatrix} 0.97 & 0.03 & 0 \\ 0.04 & 0.95 & 0.01 \\ 0.02 & 0 & 0.98 \end{bmatrix} = [\pi_C, \pi_O, \pi_I] \tag{3.5}$$

This gives with the condition $\pi_C + \pi_O + \pi_I = 1$

$$[\pi_C, \pi_O, \pi_I] = [0.52, 0.32, 0.16] \tag{3.6}$$

At equilibrium, the channel spends **52% time Closed**, **32% Open**, and **16% Inactivated**.

3.3.4 Validation by iterations

Let's start with a chosen state noted W_0 = [1,0,0]. If we fix the absolute tolerance at 10^{-4}, and using the iteration formula

$$\boldsymbol{W_{i+1} = P . W_i} \tag{3.7}$$

we get 188 states such that

$$\boldsymbol{W_{188}} \approx [0.5270, 0.3170, 0.1560] \approx [0.52, 0.32, 0.16] \tag{3.8}$$

We can increase the precision by performing more iterations (i.e., more than 188).

3.4 Inverse Estimation of a Markov Transition Matrix

3.4.1 Metropolis Algorithm

The Metropolis Monte Carlo (MC) algorithm, originally developed for simulating physical systems at thermal equilibrium (Metropolis et al., 1953), has been widely adapted for solving inverse problems through probabilistic sampling. By defining an appropriate energy or cost function related to the accuracy of a guessed transition matrix, the Metropolis algorithm allows one to stochastically explore the space of possible] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). *Optimization by Simulated Annealing*. Science, 220(4598), 671–680.

[4] matrices and converge to a good approximation of the unknown true matrix. We consider the three-state system (section 3.3) with the known transition matrix eq. (3.3), which we will attempt to rediscover solely from state trajectory data. The goal is to estimate the transition matrix using only the set of transient state vectors.

3.4.2 Energy Function

To guide the optimization of the estimated transition matrix, we define an energy (or cost or error) function that quantifies how poorly a candidate matrix reproduces the trajectory. Specifically, the energy function is

$$E(P) = \sum_{i=1}^{N-1} \| W_{i+1} - W_i . Pe \|^2 \qquad (3.9)$$

Where *Pe* is the estimated transition matrix. This is the sum of squared Euclidean distances between the actual next state (3.7) and the predicted

next state obtained using *Pe*. Minimizing this function corresponds to finding a matrix that best maps each current state to its observed successor. This approach is related to maximum likelihood estimation under Gaussian noise assumptions

3.4.3 Methodology

In our context, Metropolis algorithm is used in python code (see annex) to sample from the space of valid stochastic matrices in such a way that matrices with lower energy values are favored. The steps are as follows:

(a) Initialization:

We start with a randomly initialized matrix where each row is sampled from a uniform distribution, ensuring that rows sum to 1 and all entries are positive. An initial temperature is set to control acceptance of higher-energy moves. The temperature (which is not a physical parameter in general) is gradually reduced according to a cooling schedule controlled by a decay factor.

(b) Perturbation:

A small Gaussian perturbation is applied to a randomly chosen entry in the matrix *Pe*, followed by row normalization to maintain the stochasticity constraint (sum of each row equals 1). The perturbed matrix is accepted with probability: Accept with probability

$$Pr = \min(1, \exp(-\Delta E(P)/T)) \tag{3.10}$$

This means that when the new configuration is **better** (lower energy), we **always accept** it, and when it's **worse,** we **might still accept** it especially if the temperature is high or the energy increase is small. This helps the system **escape local minima** and explore the space better. Note that the temperature T is a control parameter in the acceptance rule that determines how tolerant the system is to unfavorable or "bad" moves during the

optimization process. When T is large, the system is more permissive, allowing most moves to be accepted, Norris, J. R. (1997). *Markov Chains*. Cambridge University Press [5] including those that worsen the objective function. This promotes exploration of the solution space. As T decreases, the system becomes more selective, increasingly favoring only those moves that improve the solution. In the limit of very small T, almost only better moves are accepted, guiding the system toward convergence.

(c) Cooling:

At each iteration, the temperature is updated from the set {10.0, 1.0, 0.1, 0.01}, thereby reducing the probability of accepting worse solutions as the optimization progresses.

(d) Convergence

The procedure runs for a fixed number of iterations (e.g., 3000), after which the matrix with the lowest recorded] Gilks, W. R., Richardson, S., & Spiegelhalter, D. (1995). *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC [6] energy is selected as the estimated transition matrix. To quantify how close each estimated matrix Pe is to the true transition matrix P, we compute the Frobenius norm of the difference between them

$$d = \sqrt{\sum_{ij}\left(P_{ij} - Pe_{ij}\right)^2} \qquad (3.11)$$

This distance measure provides a scalar value summarizing the total deviation across all matrix entries. A smaller distance implies a more accurate estimate.

3.4.4 Results

The figure 3.1 shows the energy function $E(P)$ versus the number of Metropolis iterations for four different initial temperatures: 10.0, 1.0, 0.1, 0.01. All temperature profiles demonstrate convergence of the energy function toward zero, indicating that the algorithm successfully reduces prediction error over time.

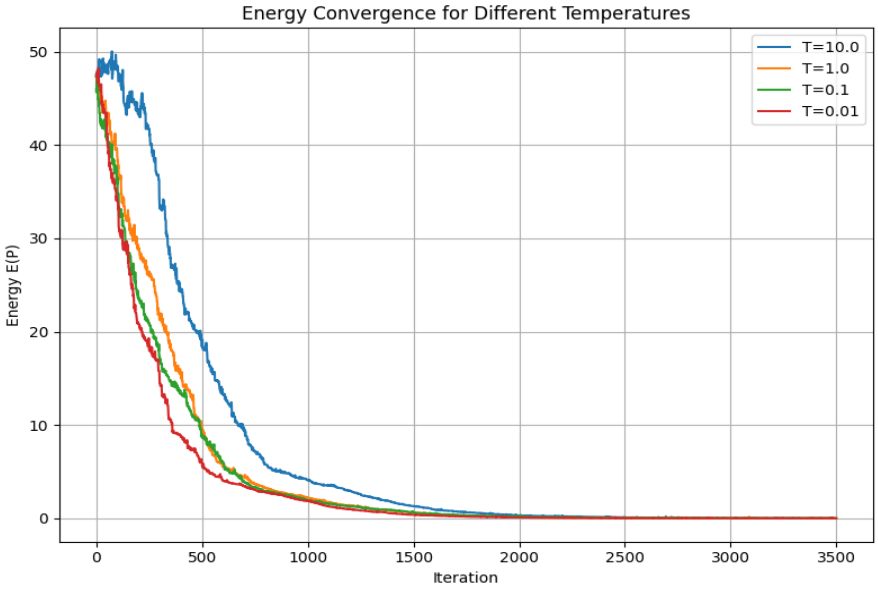

Figure 3.1Energy convergence curves

T	*D*	*p_11*	*p_12*	*p_13*	*p_21*	*p_22*	*p_23*	*p_31*	*p_32*	*p_33*
P(Exact)	00	0.97	0.03	0	0.04	0.95	0.01	0.02	0	0.98
1	0.058	0.961	0.039	0.000	0.066	0.922	0.012	0.000	0.034	0.966
0.01	0.084	0.959	0.041	0.000	0.066	0.911	0.023	0.000	0.054	0.946
10	0.101	0.951	0.049	0.000	0.077	0.897	0.026	0.008	0.055	0.937
0.1	0.108	0.957	0.043	0.000	0.066	0.901	0.033	0.008	0.067	0.925

Table 3.1 Measured distances from Pe to P

- T = 0.01 shows the fastest drop and lowest final energy, suggesting quick convergence. However, its Frobenius distance (Table 3.1) to the true matrix is not the smallest, revealing that it may have converged to a local minimum.

- T = 1.0 achieves the smallest Frobenius distance (0.058), meaning the estimated matrix is closest to the true transition matrix, even though it converges slightly slower than T = 0.01.
- T = 10.0 allows for broader exploration (due to a high acceptance rate of worse proposals), but its final distance (0.101) is worse than lower temperatures, confirming that too much randomness delays fine convergence.
- T = 0.1 shows slightly less stability than T = 1.0 and T = 0.01, ending with the worst fit (distance = 0.108), despite acceptable energy decay.

This behavior aligns with theoretical expectations from simulated annealing: moderate temperatures balance exploration and convergence well, while very high or low temperatures can lead to suboptimal estimation.

3.4.5 Conclusion

The Metropolis Monte Carlo method successfully reconstructs a Markov transition matrix from a time series of state vectors. Temperature significantly impacts the qualité and speed of convergence:

- **T = 1.0** delivers the **best compromise** between exploration and convergence, yielding the closest match to the true matrix.
- Very **low temperatures (T = 0.01)** converge fast but may **miss the global minimum**.
- Very **high temperatures (T = 10.0)** prolong exploration, potentially delaying convergence.

Hence, for similar inverse estimation problems, a **moderate initial temperature with annealing** (like T = 1.0) is recommended to ensure accurate and stable estimation.

Références

[1] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). *Equation of state calculations by fast computing machines*. Journal of Chemical Physics, 21(6), 1087–1092.

[2] Chib, S., & Greenberg, E. (1995). *Understanding the Metropolis-Hastings algorithm*. The American Statistician, 49(4), 327–335.

[3] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). *Optimization by Simulated Annealing*. Science, 220(4598), 671–680.

[4] Norris, J. R. (1997). *Markov Chains*. Cambridge University Press.

[5] Gilks, W. R., Richardson, S., & Spiegelhalter, D. (1995). *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC.

[6] Norris, J. R. (1997). *Markov Chains*. Cambridge University Press.

[7] Robert, C. P., & Casella, G. (2004). *Monte Carlo Statistical Methods* (2nd ed.). Springer.

[8] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[9] Draconis, P., & Rolles, S. W. W. (2006). *Bayesian Analysis for Reversible Markov Chains*. *Annals of Statistics*, 34(3), 1270–1292. https://doi.org/10.1214/009053606000000243

[10] Erdogdu, M. A., & Montana ri, A. (2019). *Learning Graphical Models with HMM Structure: A Bayesian Perspective*. (Preprint on arXiv). https://arxiv.org/abs/1901.02409

General Conclusion

In this study, we addressed the inverse problem of estimating the transition matrix Pe governing a three-state Markov model, representative of ion channel conformational states (e.g., open, closed, inactive). Using synthetic trajectories derived from a known matrix P, we applied the Metropolis Monte Carlo algorithm with various temperature settings to reconstruct the underlying dynamics. The energy convergence curves demonstrate that the algorithm efficiently minimizes the prediction error E(P) for all tested temperatures. However, convergence behavior and final accuracy were highly dependent on the temperature:

- **T = 1.0** yielded the **lowest Frobenius distance (D = 0.058)** from the true matrix, striking an effective balance between exploration and exploitation during sampling. The estimated transition matrix closely approximated biologically meaningful values (e.g., high self-transition probabilities for stable states).
- **T = 0.01**, although converging fastest in terms of energy, achieved only moderate reconstruction accuracy (D = 0.084), indicating that too rapid convergence can trap the algorithm in **local minima**, missing subtle dynamics—such as low-probability transitions between inactive and active states.
- **T = 10.0** enabled broader exploration but led to slower convergence and higher final error (D = 0.101), emphasizing the cost of excessive stochasticity in fine-tuning dynamic models.
- **T = 0.1** produced the **least accurate** estimate (D = 0.108), underlining that intermediate temperatures without proper annealing can perform worse than both high and low extremes.

These findings illustrate the critical role of **temperature tuning** in the Metropolis method when applied to **biophysical Markov models**. Accurate estimation of ion channel kinetics depends not only on minimizing the

energy function but also on appropriately managing exploration during sampling. In practical applications—such as fitting models to patch-clamp recordings or molecular simulations—our results recommend using a **moderate temperature with controlled annealing** to optimize convergence and matrix fidelity.

Annex: Python code

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# True transition matrix T
T_true = np.array([
    [0.97, 0.03, 0.00],
    [0.04, 0.95, 0.01],
    [0.02, 0.00, 0.98]
])

# Initial state
W0 = np.array([1.0, 0.0, 0.0])

# Generate trajectory of 188 steps
num_steps = 188
W_list = [W0]
for _ in range(num_steps - 1):
    W_next = W_list[-1] @ T_true
    W_list.append(W_next)
W_array = np.array(W_list)

# Energy function
def energy(P, W_seq):
    return sum(np.linalg.norm(W_seq[i+1] - W_seq[i] @
P)**2 for i in range(len(W_seq)-1))

# Metropolis algorithm with fixed starting matrix
```

```
def metropolis(W_seq, P_start=None, T_init=1.0,
alpha=0.99, min_temp=1e-4, max_iter=3000):
    P = P_start.copy() if P_start is not None else
np.random.dirichlet(np.ones(3), size=3)
    T = T_init
    E_list = []
    for _ in range(max_iter):
        i, j = np.random.randint(0, 3, size=2)
        P_new = P.copy()
        delta = np.random.normal(0, 0.01)
        P_new[i, j] = np.clip(P_new[i, j] + delta, 0,
1)
        P_new[i] = P_new[i] / P_new[i].sum()
        E_old = energy(P, W_seq)
        E_new = energy(P_new, W_seq)
        if E_new < E_old or np.random.rand() < np.exp(-
(E_new - E_old) / T):
            P = P_new
        E_list.append(E_new)
        T = max(T * alpha, min_temp)
    return P, E_list

# Fixed initial matrix P0
P_init = np.random.dirichlet(np.ones(3), size=3)

# Simulate for various temperatures
temperatures = [10.0, 1.0, 0.1, 0.01]
results = {}
final_matrices = {}
```

```
for T_init in temperatures:
    P_est, E_curve = metropolis(W_array, T_init=T_init,
max_iter=3500, P_start=P_init)
    results[T_init] = E_curve
    final_matrices[T_init] = P_est

# Plot energy convergence
fig1, ax1 = plt.subplots(figsize=(8, 6))
for T_init, E_curve in results.items():
    ax1.plot(E_curve, label=f"T={T_init}")
ax1.set_xlabel("Iteration")
ax1.set_ylabel("Energy E(P)")
ax1.set_title("Energy Convergence for Different
Temperatures")
ax1.legend()
ax1.grid(True)
fig1.tight_layout()
fig1.savefig("energy_curves.png")

# Create sorted table of estimated matrices
data_rows = []
for T_init, P in final_matrices.items():
    distance = np.linalg.norm(P - T_true, ord='fro')
    row = {"Temperature": T_init, "Distance to T":
round(distance, 6)}
    for i in range(3):
        for j in range(3):
            row[f"p_{i+1}{j+1}"] = round(P[i, j], 6)
    data_rows.append(row)
```

```
df = pd.DataFrame(data_rows)
df_sorted = df.sort_values(by="Distance to T")
df_sorted.to_csv("transition_matrix_table.csv",
index=False)
```