Nonequilibrium Field Theories and Stochastic Dynamics

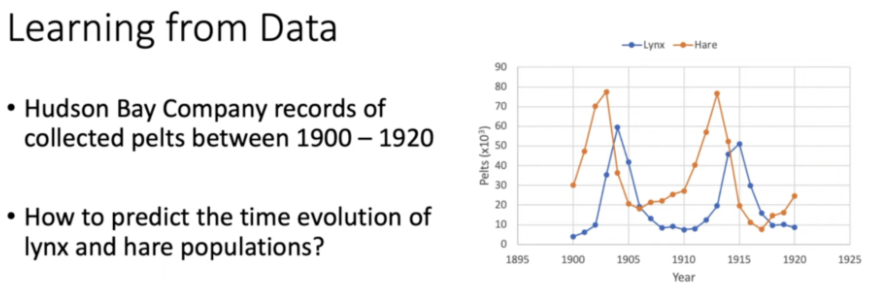
Zehong Liao

November 18, 2025

1 Problem Statement: Inferring Models from Real-World Data

One of the core tasks of scientific research is to build and validate mathematical models from observational data. A good model can not only explain existing data but also make predictions about the future. However, a model's predictive power depends entirely on the values of its parameters. So, how do we use experimental or observational data to determine these unknown parameters?

To make the issue more concrete, let's look at a classic example in ecology: the population dynamics of the Canada lynx (predator) and the snowshoe hare (prey). The Hudson Bay Company recorded the number of pelts collected from these two animals between 1900 and 1920, and this data is considered a reliable indicator of their respective population sizes.


Learning from Data

- Hudson Bay Company records of collected pelts between 1900 – 1920
- How to predict the time evolution of lynx and hare populations?

Figure 1:

By plotting this data into a chart, we can clearly see the periodic oscillations in the population size of the two species over time, and that the population peak of the predator (lynx) always lags slightly behind the population peak of the prey (snowshoe hare).

Faced with such data, a natural scientific question arises: Can we find a mathematical model to describe and predict this periodic behavior? Furthermore, can we deduce the "laws" that control the dynamics of this ecosystem from these specific data points?

To describe this kind of the predator-the prey system, a very famous mathematics model is Lotka-Volterra equations. The model uses a set of coupled ordinary differential equations (ODEs) to describe the changes in the size of the two populations.

Let x be the population size of the Snowshoe Hare (prey), and y be the population size of the Lynx (predator). The model can be written as:

$$\frac{dx}{dt} = \alpha x - \beta xy \quad \text{(Snowshoe Hare)}$$
$$\frac{dy}{dt} = -\gamma y + \delta xy \quad \text{(Lynx)}$$

Let us analyze each term in this model to understand the biological significance behind it:

- αx: This term represents the natural growth of the Snowshoe Hare. In the absence of predators ($y = 0$), the hare population grows exponentially at a rate α. Therefore, α can be viewed as the hare's **"birth rate"**.

- $-\beta xy$: This term describes the decrease in the hare population due to predation by the lynx. The frequency of predation events is proportional to the frequency of encounters between hares and lynx, which can be approximated by the product of their population sizes xy. The parameter β represents the **"success rate"** or **"predation efficiency"**.
- $-\gamma y$: This term represents the natural death of the lynx. In the absence of food (hares, $x = 0$), the lynx population decays exponentially at a rate γ due to starvation, disease, or intraspecific competition. Therefore, γ can be viewed as the lynx's **"death rate"**.
- $+\delta xy$: This term describes the growth of the lynx population gained from preying on hares. Similarly, this growth rate is also proportional to the encounter frequency xy. The parameter δ represents the **"conversion efficiency"** of turning consumed hares into lynx reproduction.

Now, our problem becomes more precise. The "rules" describing the dynamics of this system are encapsulated in these four parameters $\theta = \{\alpha, \beta, \gamma, \delta\}$. Our task is: given observed data D (i.e., that time-series chart), how do we infer the most probable values for this set of parameters θ? This is the core problem of **parameter inference**.

This process represents a leap from chaotic real-world observations (fur data) to abstract mathematical models (differential equations). The Lotka-Volterra model is undoubtedly a simplification of complex ecosystems, but it captures the core feedback loop of the predator-prey interaction. The challenge lies not only in solving this set of equations, but also in finding a specific "version" of the equations (i.e., the most suitable parameter values) that best represents the real world we observe. This lays the foundation for our subsequent introduction of probabilistic methods.

How can we systematically learn parameters from data? Bayesian inference provides a logically rigorous framework based on probability theory to solve this problem. Essentially, it's a method for updating our beliefs about things after acquiring new evidence

The core tool of Bayesian inference is Bayes' Theorem. It connects the quantities of interest and takes the following form:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \tag{1}$$

Although this formula appears simple, it embodies profound logic. It tells us how to combine **prior knowledge** about parameters with the information contained in the data, thereby obtaining **updated knowledge**.

To apply this theorem to our Lynx-Snowshoe Hare problem, we need to understand every component of the formula.

The perspective of Bayesian inference is fundamentally different from traditional "best fit" methods (such as Least Squares). Traditional methods might yield a unique set of "best" parameter values. However, the Bayesian method tells us: "There is no single correct θ. Instead, there is a landscape of possibilities. This region in the parameter space is highly probable, while that region is highly unlikely."

This probabilistic perspective is very powerful. It allows us to answer deeper questions, for example: "What is the probability that the Snowshoe Hare birth rate α lies between 0.5 and 0.6?" or "Are the predation rate β and the Lynx death rate γ correlated?" **This represents a profound shift from deterministic to probabilistic thinking, which more honestly reflects the uncertainty we face when dealing with limited data.**

Having a posterior distribution theoretically solves the inference problem, but in practice, we usually need to extract some summary information from this distribution, such as the expected value (mean) or variance (the magnitude of uncertainty) of the parameters.

To calculate the expected value of a certain function of the parameters $f(\theta)$ (for example, $f(\theta) = \alpha$ is used to calculate the mean value of parameter α), we need to solve the following integral:

$$\langle f(\theta) \rangle = \int f(\theta) p(\theta|D) d\theta^n \tag{2}$$

Here n is the number of parameters (in our example, $n = 4$). This integral is performed over the entire n-dimensional parameter space.

Furthermore, we also need to calculate the normalization constant, known as the evidence $p(D)$, which is itself an integral of the same form:

$$p(D) = \int p(D|\theta)p(\theta)d\theta \tag{3}$$

This integral is usually the most difficult part of Bayesian inference.

Why are these integrals so difficult to calculate? The reason lies in the so-called **Curse of Dimensionality**.

One of the most intuitive methods for numerical integration is the "grid method": we divide each parameter dimension into several discrete grid points, then calculate the value of the integrand at these points, and finally perform a weighted sum.

Let us see why this method fails rapidly. Suppose we take only 10 grid points for each parameter dimension.

- For 1 parameter (1D), we need to calculate $10^1 = 10$ times.
- For 2 parameters (2D), we need to calculate $10^2 = 100$ times.
- For our 4-parameter Lotka-Volterra model (4D), we need to calculate $10^4 = 10,000$ times.
- For a more complex model with 20 parameters, we need to calculate 10^{20} times! This number far exceeds the processing power of any modern computer.

The computational cost grows exponentially with dimensionality, making the grid method completely infeasible in high-dimensional spaces.

A deeper problem is that the behavior in high-dimensional space runs counter to our low-dimensional intuition. As shown in the teaching assistant's sketch on the blackboard, the "mass" of the posterior probability distribution $p(\theta|D)$ is usually concentrated in a very tiny, irregularly shaped region within the parameter space. Whereas in the vast majority of the extensive parameter space, the probability density is nearly zero.

2 Monte Carlo Integration

To address the challenges of high-dimensional integration, we need a smarter approach. Monte Carlo integration offers a brilliant solution. It replaces deterministic grid partitioning with random sampling.

The core idea of Monte Carlo integration is based on the Law of Large Numbers. It approximates the computation of the integral as a sample mean calculation:

$$\langle f(\theta) \rangle = \int f(\theta)p(\theta|D)d\theta \approx \frac{1}{N}\sum_{i=1}^{N} f(\theta^{(i)}) \tag{4}$$

This formula looks very simple, just like calculating the ordinary average of $f(\theta)$. But its magic lies in how these samples $\theta^{(i)}$ are generated.

To make this approximation effective and efficient, these samples $\theta^{(i)}$ must be **drawn from the target probability distribution $p(\theta|D)$ itself.**

This is the core idea of "Importance Sampling". By sampling directly from $p(\theta|D)$, we naturally concentrate computational effort on regions with high probability density—that is, the regions that contribute most to the integral. We no longer waste time in the "deserts" of zero probability within the parameter space. Sample points will appear with a frequency proportional to their probability, which is exactly what we need.

This shift is revolutionary. We have successfully transformed a intractable integral problem into an equally challenging but solvable sampling problem:

How do we generate random samples from a complex, high-dimensional probability distribution $p(\theta|D)$ that we cannot even normalize (because we do not know $p(D)$)?

Monte Carlo integration represents a paradigm shift from deterministic, exhaustive calculation to approximate, stochastic simulation. It embraces randomness and uses it as a powerful tool to overcome the Curse of Dimensionality. It solves the efficiency problem because we no longer need to know in advance where the important regions are; the sampling process automatically discovers them. Now, what we need is no longer an integrator, but a **sampler**. This provides the ultimate motivation for us to introduce the entire theoretical system of stochastic processes.

2.1 MCMC's core idea

Now, we reveal the core idea of the MCMC method: **What if we could design a Markov Chain such that its unique stationary distribution π happens to be exactly the target posterior distribution $p(\theta|D)$ that we want to sample from?**

If we can achieve this, the sampling process becomes surprisingly simple:

1. Randomly select an initial point $\theta^{(0)}$ in the parameter space.
2. Let this Markov Chain continuously transition states (Random Walk) according to its transition rules.
3. Run it for a sufficiently long time to let the chain "forget" its initial state and converge to its stationary distribution. This initial phase is called the **"burn-in"** period.
4. After the burn-in period ends, we begin recording the sequence of visited states $\{\theta^{(i)}\}$.

According to the definition of the stationary distribution, this sequence $\{\theta^{(i)}\}$ is exactly the series of samples drawn from the target distribution $p(\theta|D)$ that we have been dreaming of! We have successfully built a sampler.

This is the conceptual core of this lecture. MCMC transforms descriptive stochastic process theory into a prescriptive, actionable computational engine. We are no longer merely analyzing the equilibrium state of a given physical system; we are **engineering** a calculation "system" (a Markov Chain) so that its equilibrium state becomes a useful tool for us (the posterior distribution).

Previous lectures taught us: "If you have a transition matrix W, here is how to find its stationary distribution π." Whereas MCMC turns this problem completely upside down: **"We have a desired target stationary distribution $\pi = p(\theta|D)$, how can we find a transition matrix W that generates it?"** This is a leap from analysis to synthesis. Markov Chain theory (especially the detailed balance condition) provides the blueprint for constructing such a W.

2.2 Metropolis-Hastings Algorithm: A Practical Recipe

So, how exactly do we construct a transition rule W that satisfies the conditions? The **Metropolis-Hastings Algorithm** provides a general and powerful recipe to construct a Markov chain for any target distribution.

The Metropolis-Hastings algorithm is one of the most important algorithms in modern computational statistics and physics, spanning two key innovations in the mid-20th century. The original version of this algorithm was proposed in 1953 by Nicholas Metropolis and his colleagues (including Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller, and Edward Teller). At the time, it was primarily used to solve problems regarding equation of state calculations for high-dimensional systems in physics. Its core idea was to construct a Markov chain such that its final stationary distribution is the target distribution we wish to sample. The initial version was limited to symmetric proposal distributions. Subsequently, in 1970, statistician W. K. Hastings generalized the algorithm to include non-symmetric proposal distributions, greatly expanding its scope and forming the Metropolis-Hastings algorithm we know today.

The application of this algorithm is extremely widespread, playing a core role especially in Bayesian statistics. When a model's posterior probability distribution is too complex or high-dimensional for direct analytic calculation or sampling, the Metropolis-Hastings algorithm provides a powerful numerical simulation tool. Specific applications include but are not limited to: Bayesian inference in machine learning for estimating complex model parameters; constructing phylogenetic trees in computational biology; simulating the behavior of multi-particle systems in physics; and risk modeling and option pricing in finance. It is fair to say that the Metropolis-Hastings algorithm can be found in any scientific or engineering field that requires sampling from a probability distribution that is difficult to handle directly.

The workflow of the algorithm is very clear and can be divided into two stages: **Propose** and **Accept/Reject**.

1. Initialization: Arbitrarily select an initial state $\theta^{(0)}$ in the parameter space.

2. Iteration: For $i = 1, 2, \ldots, N$:

a. Propose: Generate a candidate state θ' based on a proposal distribution $q(\theta'|\theta^{(i-1)})$ starting from the current state $\theta^{(i-1)}$. This proposal distribution can be very simple, for example, a normal distribution centered at the current state.

b. Calculate Acceptance Rate (Accept/Reject): Calculate the probability of accepting this proposal $A(\theta'|\theta^{(i-1)})$. The acceptance probability given by the Metropolis-Hastings algorithm is:

$$A(\theta'|\theta) = \min\left(1, \frac{p(\theta')q(\theta|\theta')}{p(\theta)q(\theta'|\theta)}\right) \tag{5}$$

Note: Here we use $p(\theta)$ to represent our target distribution $p(\theta|D)$.

c. Decision: Draw a random number u from the uniform distribution $[0, 1]$.

If $u < A$, accept this proposal, set the new state $\theta^{(i)} = \theta'$.

Otherwise, reject this proposal, set the new state $\theta^{(i)} = \theta^{(i-1)}$ (i.e., stay in the same place).

By repeating this process, we obtain a Markov Chain sample sequence $\{\theta^{(0)}, \theta^{(1)}, \ldots, \theta^{(N)}\}$.

The ingenuity of this algorithm lies in the fact that the defined acceptance rate A precisely ensures that the entire process satisfies the *detailed balance condition*, thereby guaranteeing that the stationary distribution of the chain is our target distribution $p(\theta)$. Let us understand the two key ratios in the acceptance rate formula:

- $\frac{p(\theta')}{p(\theta)}$: This is the ratio of the probability densities of the target distribution at the proposed point and the current point. If the proposed new point θ' lies in a region of higher probability ($p(\theta') > p(\theta)$), then this ratio is greater than 1, the acceptance rate A becomes 1, and we always accept this move. This ensures the chain tends to move towards high-probability regions. If the proposed point has lower probability, we accept it with a certain probability, which allows the chain to explore the entire distribution, not just stay at the highest peak of probability.

- $\frac{q(\theta|\theta')}{q(\theta'|\theta)}$: This is the correction factor for the proposal distribution. It is used to correct any potential "proposal bias". If proposing θ' from θ is easier than proposing θ from θ', this ratio compensates for it, ensuring the exploration is fair. In many practical applications, we choose a **symmetric proposal distribution**, such as the normal distribution, where $q(\theta|\theta') = q(\theta'|\theta)$, and this correction factor equals 1. The algorithm simplifies to the original **Metropolis Algorithm**, with an acceptance rate of $A = \min(1, \frac{p(\theta')}{p(\theta)})$.

This algorithm has another huge practical advantage: Note that in the calculation of the acceptance rate, we only need the **ratio** of the target distribution $\frac{p(\theta')}{p(\theta)}$. This means that for Bayesian inference $p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$, that extremely difficult-to-calculate normalization constant (evidence) $p(D)$ is perfectly cancelled out in the ratio!

$$\frac{p(\theta|D)}{p(\theta'|D)} = \frac{p(D|\theta)p(\theta)/p(D)}{p(D|\theta')p(\theta')/p(D)} = \frac{p(D|\theta)p(\theta)}{p(D|\theta')p(\theta')} \tag{6}$$

We only need to be able to calculate the likelihood $p(D|\theta)$ and the prior $p(\theta)$, which is usually feasible. This is one of the key reasons for the success and popularity of MCMC methods. It cleverly bypasses the biggest computational obstacle in Bayesian inference.

The structure of the algorithm can be viewed as an intelligent "Propose-Correct" system. The **proposal distribution** q is responsible for exploration; it can be simple or even "blind". The **acceptance probability** A acts as an intelligent filter. By enforcing the physical law of detailed balance, it ensures that no matter how the exploration process proceeds, the final result will converge to the correct target distribution.

3 Solving the parameters of the Lotka-Volterra model using MCMC

Now, we put theory into practice. We return to the initial example of the lecture: the population dynamics of the Lynx and the Snowshoe Hare. We will use the Metropolis-Hastings MCMC algorithm, based on data provided by the Hudson's Bay Company, to infer the four parameters $(\alpha, \beta, \gamma, \delta)$ of the Lotka-Volterra model that most likely describe this ecosystem.

The challenge of this case lies in the fact that the Lotka-Volterra model is a system of differential equations with no simple analytic solution. For a given set of parameters, we need to obtain the population evolution curves through numerical integration. This makes the posterior probability distribution $p(\theta|D)$ extremely complex, rendering direct calculation or sampling impossible. This is precisely the scenario where MCMC excels.

1. **Model:** Lotka-Volterra system of differential equations.
2. **Data:** Population counts of Snowshoe Hare (H) and Lynx (L) from 1900-1920.
3. **Objective:** Solve for the posterior probability distribution $p(\alpha, \beta, \gamma, \delta|\text{Data})$.
4. **Method:** Construct a Metropolis-Hastings sampler.
    - **Likelihood Function $p(D|\theta)$:** We assume the error between observed data and model predictions follows a log-normal distribution. This means we calculate the predicted population counts under given parameters θ, compare them with the real data, and calculate the probability.
    - **Prior Distribution $p(\theta)$:** Since we do not have much prior knowledge about the parameters, we choose an uninformative, broad uniform distribution or normal distribution as the prior.
    - **State Space:** 4-dimensional parameter space $(\alpha, \beta, \gamma, \delta)$.
    - **Target Distribution:** Posterior probability $p(\theta|D) \propto p(D|\theta)p(\theta)$.
    - **Proposal Distribution $q(\theta'|\theta)$:** We adopt a simple random walk strategy. Near the current parameter point θ, we generate a new proposal point θ' by adding a small random perturbation (e.g., sampling from a multivariate normal distribution).

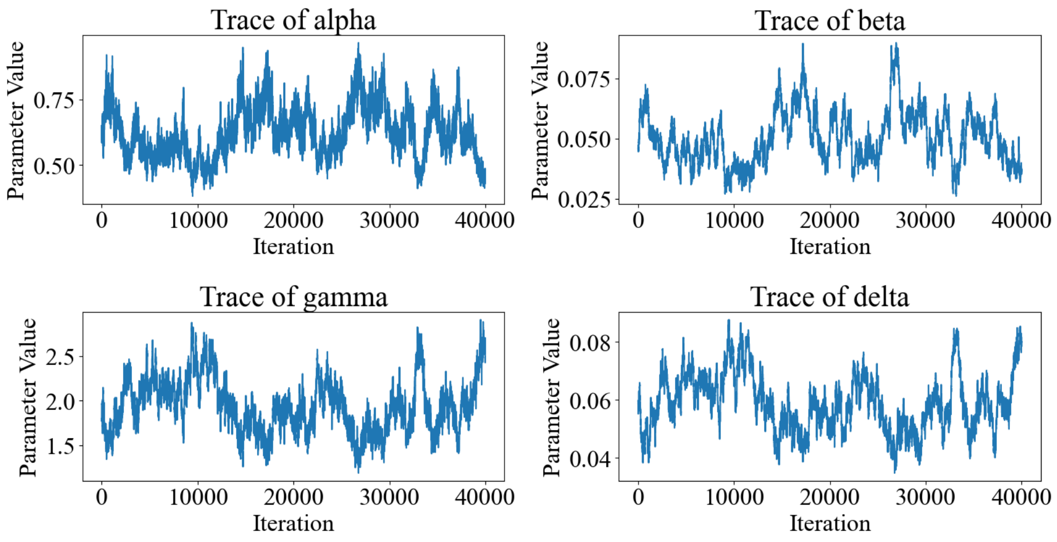


Figure 2:

Trajectory Plot: This plot shows how the sampled values of each parameter change during the iterations of MCMC. Ideally, we want to see these trajectories resemble a ”caterpillar,” randomly fluctuating around a stable value without a clear upward or downward trend. This indicates that the Markov chain has ”forgotten” its initial position and has begun exploring the typical region of the target posterior distribution. If the trajectory plot shows a clear trend, it usually means that the ”burn-in” phase is not long enough, or the proposed distribution is inappropriate, resulting in low mixing efficiency of the chain.

Posterior Distributions: These are the core results of our Bayesian inference. These histograms approximate our perception of the uncertainty of each parameter given the data. The peak of the

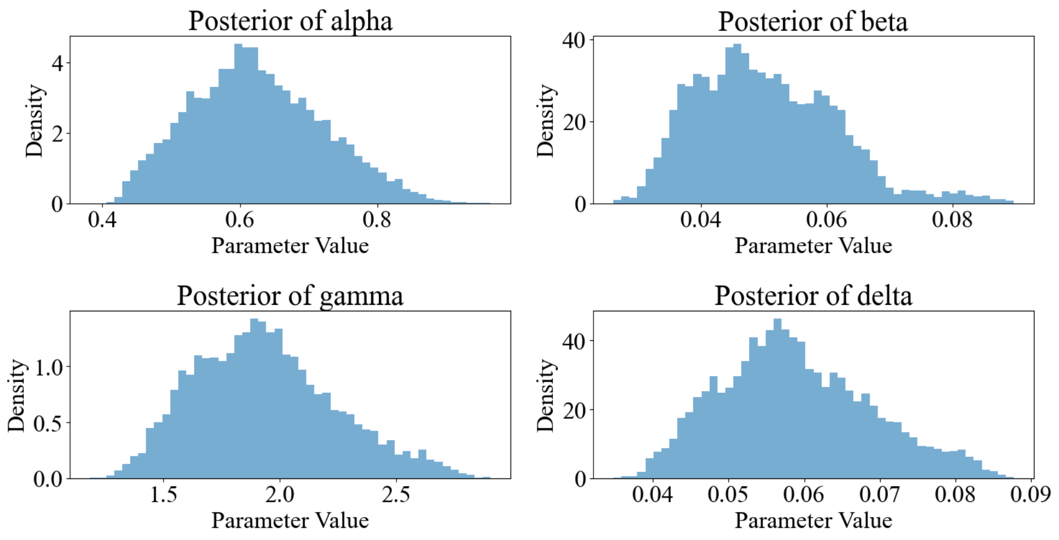


Figure 3:

distribution represents the most likely value of the parameter (i.e., the maximum a posteriori estimate), while the width of the distribution quantifies our uncertainty about that value. For example, a tall and narrow distribution means that the data strongly constrains the parameter to a small range; while a short and wide distribution indicates that the data provides limited information, and we are still not entirely certain about the true value of the parameter.

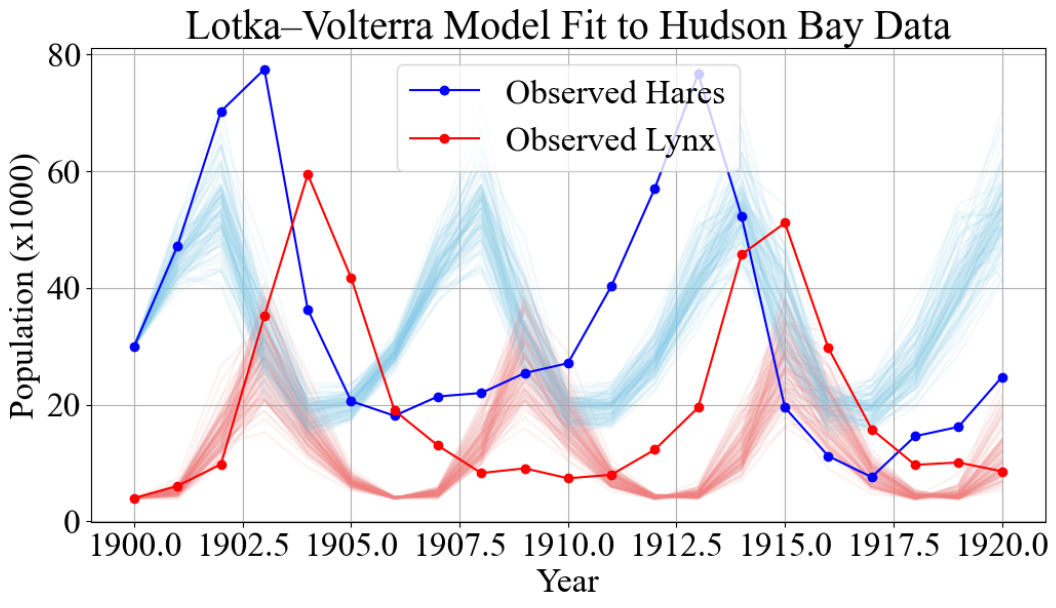


Figure 4:

Model Prediction vs. Data Comparison: This chart visually compares our inferences with real-world data. The semi-transparent thin lines represent the model evolution trajectories resulting from 100 sets of different parameters randomly selected from the posterior distribution. The ”confidence bands” formed by these curves show the model's predictive range after considering parameter uncertainties. We can see that the actual data points (blue and red dots) mostly fall within this confidence band, indicating that our Lotka-Volterra model and the inferred parameters can effectively capture and reproduce the historical periodic fluctuations in lynx and snowshoe hare populations.This provides strong visual evidence for the model's effectiveness.

Through this case, we have fully explored the process of using advanced sampling methods to solve practical scientific problems: starting with a real-world dataset, building a mathematical model, defining the target (posterior distribution) using a Bayesian inference framework, and finally, by constructing a clever stochastic process (MCMC), successfully drawing samples from this complex target distribution, thus completing the quantitative learning of the model parameters. This precisely demonstrates the powerful effectiveness of the Monte Carlo method as a stochastic process in modern scientific research.

4 "Random Walk" Metropolis-Hastings and Its Fatal Flaws

A very common symmetric proposal distribution is a Gaussian distribution centered at the current state:

$$q(\theta'|\theta) = \mathcal{N}(\theta'|\theta, \sigma^2) \tag{7}$$

This method is intuitive and easy to implement, but it is exactly the root cause of the algorithm's inefficiency. The performance of this random walk heavily depends on the choice of the step size σ.

Small step size σ: The proposed new state will be very close to the current state. Since the change in $p(\theta)$ is small, the acceptance rate will be high. However, the chain moves very slowly, exploring the distribution in a diffusive manner. This leads to high autocorrelation between samples, meaning we need a very long chain to obtain a representative set of independent samples. **The algorithm might get "stuck" in local probability peaks.**

Large step size σ: In principle, the algorithm can explore the space faster. However, in any slightly complex distribution, a large random jump is highly likely to land in a region with much lower probability (a distribution's "typical set" is usually a thin shell, not a solid ball). This leads to a sharp drop in the acceptance rate α, causing the chain to stagnate, rejecting almost all proposals.

This dilemma of step size selection becomes particularly severe in high-dimensional spaces. In 1-dimensional space, a random step has a 50% chance of heading in the generally "correct" direction. But in D-dimensional space, the volume of the state space grows exponentially. A random step is almost certain to point in a "bad" direction, away from the narrow region where high probability lies. Therefore, as the number of parameters D increases, the efficiency of the random walk drops exponentially.

This trade-off makes the Random Walk Metropolis-Hastings sampler unsuitable for complex, high-dimensional probability distributions often encountered in modern Bayesian statistics, machine learning, and physics. We need a method to propose new states that are both **far away** and **likely to be accepted**.

To overcome the fundamental limitation of the random walk's inefficiency in high-dimensional space, we need a "smarter" exploration strategy. The inspiration for this strategy comes from one of the cornerstone models of statistical physics—the **Ising model**.

The Ising model is a paradigm for understanding phase transitions (such as the magnetization of magnets). In this model, the probability that the system is in a specific microstate (for example, a sequence of spins up or down, denoted by $\{\sigma\}$) is entirely determined by its energy and the ambient temperature, taking the form of the elegant **Boltzmann distribution**:

$$p(\{\sigma\}) = \frac{1}{Z} e^{-\beta H(\{\sigma\})} \tag{8}$$

Here, H is the Hamiltonian of the configuration (that is, the energy), β is a constant related to temperature, and Z is the normalization constant (partition function) that ensures the total probability sums to 1. This formula embodies a profound physical intuition: **the lower the energy of a state, the exponentially higher its probability of appearing.** Physical systems always tend to stay in more stable, low-energy states.

This physical picture gives us immense inspiration. Can we "flip" this logic: for any abstract target probability distribution $p(\theta)$ we want to sample, can we define an equivalent "energy" function $E(\theta)$ for it? The answer is yes, we can enforce them to satisfy a relationship similar to the Boltzmann distribution:

$$p(\theta) \propto e^{-E(\theta)} \tag{9}$$

By taking the logarithm of both sides, we can directly write down the conversion relationship between them:

$$E(\theta) = -\log p(\theta) + \text{const} \tag{10}$$

We call this newly defined $E(\theta)$ the **Potential Energy**.

This seemingly simple mathematical transformation is actually a fundamental cognitive leap; it completely changes the way we view sampling problems.

1. **From "Blind" to "Directed":** Our initial goal was to find regions with high $p(\theta)$ in high-dimensional space. Now, our equivalent new goal is to explore regions with low potential energy $E(\theta)$. This successfully transforms a pure statistical sampling problem **into a problem rich in physical intuition: exploring an energy landscape.** What was originally a high-probability "peak" in $p(\theta)$ now becomes a deep "valley" in the $E(\theta)$ energy landscape.

2. **Introducing the concept of "Force":** The reason random walks are inefficient is that they are "blind"; they have no idea about the terrain when proposing the next step. But in the physical world, the motion of objects is not blind. A small ball rolling in a valley is subject to forces. This force is exactly the **negative gradient** of the potential energy:

$$F = -\nabla E(\theta) = -\nabla(-\log p(\theta)) = \nabla \log p(\theta) \tag{11}$$

We have found the key to shedding randomness. We now have a "force" given directly by the logarithmic gradient of the target distribution, and this force **always points in the direction where probability density increases fastest.** We no longer need blind trial and error; we can follow the direction of the "force" to efficiently advance towards high-probability regions.

Through this perspective shift, we concretize the abstract sampling problem into simulating the motion of a physical particle in a potential energy field defined by the target distribution. This particle will be naturally "pushed" towards the regions we are most interested in.

The idea of applying the dynamics of physical systems to statistical computing first fermented in the physics community. The seminal work that crystallized this idea into what we know today as Hamiltonian Monte Carlo (then called Hybrid Monte Carlo) was published by **Duane, Kennedy, Pendleton, and Roweth in 1987**. Their goal at the time was to solve challenging high-dimensional integration problems in Lattice Quantum Chromodynamics (Lattice QCD). Later, this powerful method was discovered and popularized by the statistics and machine learning communities (especially by **Radford Neal**), eventually becoming one of the most efficient and core algorithms in modern Bayesian inference.

By introducing the concept of "potential energy," the sampling problem is successfully transformed into an exploration of a physical energy landscape. However, this is not enough, because a "particle" with only position and potential energy is stationary and cannot explore on its own. To make the particle "move," we need to give it momentum. This is the core innovation of Hamiltonian Monte Carlo (HMC) and the fundamental difference between it and traditional MCMC methods. The complete idea of HMC did not come from nowhere. Its history can be traced back to the 1980s, when it was first proposed by physicists such as Simon Duane in solving complex calculations of lattice gauge theory. They ingeniously realized that the Hamiltonian dynamics framework used in classical mechanics to describe planetary motion could be borrowed to guide the statistical sampling process, thereby changing its efficiency.

In classical mechanics, the total energy of an isolated system is conserved. This total energy, namely the **Hamiltonian** H, is the sum of its potential and kinetic energy. We adopt this core concept from physics exactly to construct our system:

$$H(\theta, v) = E(\theta) + K(v) \tag{12}$$

- **Potential Energy $E(\theta)$:** As defined above, $E(\theta) = -\log p(\theta)$. This part is determined entirely by our target probability distribution; it constructs the "terrain" or "energy landscape" of the particle's motion. Regions with higher probability have lower potential energy.

- **Kinetic Energy $K(v)$:** This is the energy associated with motion. Its form is directly borrowed from physics, a quadratic function of momentum:

$$K(v) = \frac{1}{2} v^T M^{-1} v \tag{13}$$

Here M is a symmetric positive-definite matrix, called the **Mass Matrix**.

  - **Simple Case:** The simplest choice is to set M as the **Identity Matrix** ($M = I$). This means we assume the "mass" of all parameter dimensions is the same, and the particle's response to "force" is equal in all directions.
  - **Deeper Implication:** In more advanced applications, M can be set to be related to the covariance of the target distribution. This makes the particle have larger "mass" (harder to push, moving more cautiously) in directions where the probability density is narrow, and smaller "mass" (easier to push, moving more boldly) in directions where the probability density is broad, thereby greatly improving sampling efficiency on complex correlated distributions.

- **Momentum v:** Can be intuitively understood as the "speed and direction" of the particle exploring the parameter space.
- **Mass Matrix M:** In physics, mass represents the magnitude of inertia. Here, it plays an important regulatory role.

We now define a **joint probability distribution** $p(\theta, v)$ regarding position and momentum based on this Hamiltonian, in the exact same form as the Boltzmann distribution, which is known in physics as the **Canonical Distribution**:

$$p(\theta, v) = \frac{1}{Z'} e^{-H(\theta,v)} = \frac{1}{Z'} e^{-(E(\theta)+K(v))} = \frac{1}{Z'} e^{-E(\theta)} e^{-K(v)} \tag{14}$$

This construction seems to make the problem more complicated: we originally only wanted to sample from $p(\theta)$, but now we have to sample from a higher-dimensional $p(\theta, v)$. The secret here is that the structure of this joint distribution is **separable**. Because the Hamiltonian is designed as the sum of two parts, the joint probability can be **factorized**:

$$p(\theta, v) = \left(\frac{1}{Z_E} e^{-E(\theta)} \right) \left(\frac{1}{Z_K} e^{-K(v)} \right) = p(\theta) p(v) \tag{15}$$

Under our common choice for kinetic energy $K(v)$, $p(v)$ is simply a centered Gaussian distribution: $p(v) = \mathcal{N}(v|0, M)$. This factorization is the cornerstone of the entire HMC theoretical framework. It guarantees us that:

In this extended joint distribution, position θ and momentum v are mutually independent. This means that although the momentum variable we introduced is crucial in the **dynamical evolution** (it drives the exploration), it does not affect the **static target distribution** we ultimately sample. Therefore, if we can design an algorithm to generate sample pairs (θ, v) from the joint distribution $p(\theta, v)$, we only need to **simply discard the momentum v component** after sampling. The remaining θ is the effective sample from the original target distribution $p(\theta)$ that we have been dreaming of.

Momentum is like a temporary "booster" or "scaffold": at the beginning of each step, we randomly set a momentum (giving the particle a random "kick"), let the system evolve under the guidance of physical laws, thereby efficiently proposing a high-quality candidate point. After the task is completed, we throw away this booster and prepare for the next launch.

5 Hamilton Monte Carlo Step-by-Step Guide

At the beginning of each iteration, we discard the old momentum and draw a new one from its distribution $p(v)$. This is usually a standard multivariate Gaussian distribution:

$$v \sim \mathcal{N}(0, M) \tag{16}$$

From the perspective of physical intuition, this step is equivalent to giving our fictitious particle a random "kick". This changes the total energy H of the system, allowing the sampler to jump between different energy contours, thereby exploring the entire energy landscape. Without this step, the simulation would be purely deterministic and trapped on a single energy iso-surface.

Given a starting point $(\theta, \tilde{v})$, we generate a proposal (θ', v') by evolving the system forward for a fixed duration T according to the Hamiltonian equations of motion:

$$\begin{aligned} \frac{d\theta}{dt} &= \frac{\partial H}{\partial v} = M^{-1}v \\ \frac{dv}{dt} &= -\frac{\partial H}{\partial \theta} = -\nabla_\theta E(\theta) = \nabla_\theta \log p(\theta) \end{aligned} \tag{17}$$

From the perspective of physical intuition:

- The first equation is simple: the rate of change of position is velocity ($M^{-1}v$).
- The second equation is Newton's Second Law ($F = ma$): the rate of change of momentum is force, i.e., the negative gradient of potential energy. The particle is "pulled" towards regions of lower energy (higher probability).

This is exactly the core of HMC's power. Unlike a random walk, its trajectory is guided by the gradient of the log-probability. The particle will naturally move towards and explore high-probability regions. This enables us to propose a new state θ' that is far away from the current state θ yet still resides in a reasonable region, thereby obtaining a very high acceptance rate.

The Hamiltonian equations form a continuous system of differential equations. For most problems of interest, we cannot solve them analytically, so we must use numerical integrators to approximate the trajectories. Simple methods like Euler's method rapidly accumulate errors and fail to conserve energy, leading to very low acceptance rates. HMC requires a special type of integrator called a **symplectic integrator**. The most common choice is the **Leapfrog integrator**.

The Leapfrog algorithm works by first updating the momentum by a half-step, then updating the position by a full step, and finally updating the momentum by another half-step. For a small time step ϵ, a Leapfrog step proceeds as follows:

1. $v(t + \epsilon/2) = v(t) - (\epsilon/2)\nabla_\theta E(\theta(t)) = v(t) + (\epsilon/2)\nabla_\theta \log p(\theta(t))$
2. $\theta(t + \epsilon) = \theta(t) + \epsilon M^{-1} v(t + \epsilon/2)$
3. $v(t + \epsilon) = v(t + \epsilon/2) - (\epsilon/2)\nabla_\theta E(\theta(t + \epsilon)) = v(t + \epsilon/2) + (\epsilon/2)\nabla_\theta \log p(\theta(t + \epsilon))$

To simulate a total time T, we repeat this process $L = T/\epsilon$ times.

Why go to the trouble of using this specific update scheme? Symplectic integrators like Leapfrog possess two crucial properties:

- **Time Reversibility:** If you run L steps of Leapfrog from (θ, v) to reach (θ', v'), then reverse the final momentum to $-v'$ and run L steps again, you will return exactly to $(\theta, -v)$. This property is crucial for satisfying detailed balance.
- **Volume Preservation:** The integrator preserves the volume of the (θ, v) phase space. This means it does not artificially compress or dilute probability density, which is necessary for a valid MCMC algorithm.

Although Leapfrog cannot perfectly conserve the Hamiltonian H (it has small errors of the order ϵ^2), it possesses excellent long-term stability and the geometric properties required to maintain the validity of MCMC transitions. This allows HMC to maintain a very high acceptance rate even after many integration steps.

After L steps of Leapfrog integration, we obtain a proposal state (θ', v'). Since numerical integration is not perfect, the total energy H is not perfectly conserved. $H(\theta', v')$ will differ slightly from $H(\theta, \tilde{v})$. To correct this numerical error and make the algorithm exact, we perform the final Metropolis acceptance step (1):

$$\alpha = \min\left(1, \exp\left[-(H(\theta', v') - H(\theta, \tilde{v}))\right]\right) \tag{18}$$

This point is very profound. Unlike Random Walk Metropolis, the accept/reject step in HMC is not the primary driver of exploration. **Hamiltonian dynamics is the explorer.** The acceptance step is a **quality control check** on the numerical integrator. If the integrator performs well in conserving energy ($H' \approx H$), then the acceptance probability α will be close to 1, just as written on the blackboard

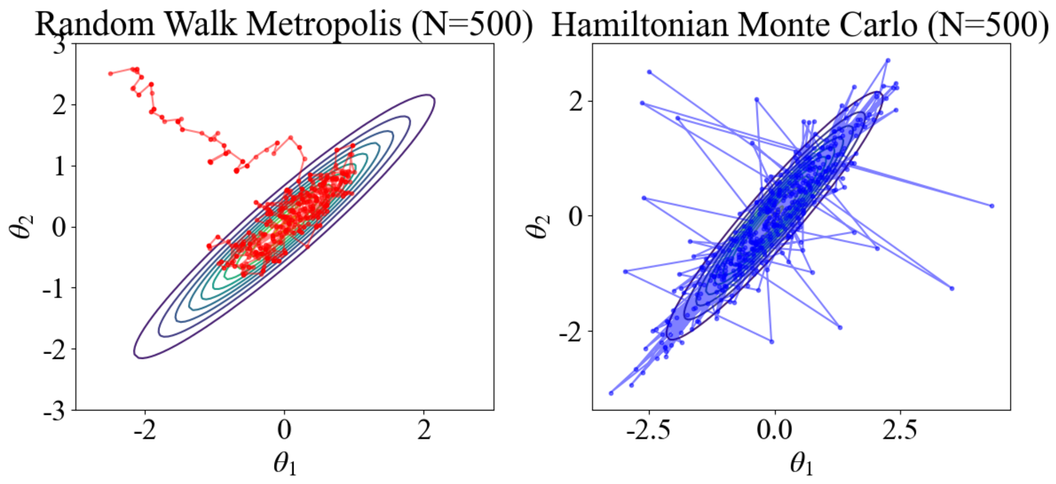


Figure 5:

($\alpha \approx 1$). If the step size ϵ is too large, the integration error will be large, energy will not be conserved, and the proposal will be (correctly) rejected.

As can be clearly seen from the figures, the RWM sampler (left figure) has a very dense path, small steps, and slow exploration speed, exhibiting typical diffusion behavior. In contrast, the HMC sampler (right figure) can make long-distance jumps, efficiently moving between high-probability regions of the distribution, thus obtaining representative samples of the entire distribution more quickly.

6 Solving the parameters of the Lotka-Volterra model using modern Bayesian methods

Now, we put theory into practice and reproduce the example shown in the lecture. Returning to the initial example of the lecture: the population dynamics of the Lynx and Snowshoe Hare. We will use the Hamiltonian Monte Carlo (HMC) algorithm, within a modern probabilistic programming framework called **PyMC**, based on data provided by the Hudson's Bay Company, to infer the seven unknown parameters of the Lotka-Volterra model describing this ecosystem ($\alpha, \beta, \gamma, \delta$, initial population counts H_0, L_0, and observation noise σ).

The challenge of this case lies in the fact that the Lotka-Volterra model is a system of differential equations with no simple analytic solution. For a given set of parameters, we need to obtain the population evolution curves through numerical integration. This makes the posterior probability distribution $p(\theta|D)$ extremely complex and relatively high-dimensional, with potentially strong correlations between parameters. This is precisely the scenario where advanced MCMC algorithms like HMC excel.

1. **Model:** Lotka-Volterra system of Ordinary Differential Equations (ODEs).
2. **Data:** Population counts of Snowshoe Hares and Lynx from 1900-1920.
3. **Objective:** Solve for the joint posterior probability distribution of all unknown parameters $p(\alpha, \beta, \gamma, \delta, H_0, L_0, \sigma|\text{Data})$.
4. **Method:** Construct a probabilistic model in PyMC and use its default NUTS (No-U-Turn Sampler), which is an efficient variant of HMC.

   - **State Space:** 7-dimensional parameter space $(\alpha, \beta, \gamma, \delta, H_0, L_0, \sigma)$.
   - **Target Distribution:** Posterior probability $p(\theta|D) \propto p(D|\theta)p(\theta)$.
   - **Likelihood Function $p(D|\theta)$:** We assume the error between observed data and model predictions follows a log-normal distribution. This means the population counts predicted

by the model (which must be positive) have normal errors relative to the log of observed data on a logarithmic scale. This is a very reasonable assumption for population data that cannot be negative and where variance might scale with the mean.

- **Prior Distribution $p(\theta)$:** We selected Log-Normal or Half-Normal distributions as priors for all parameters. This is a key modeling choice aimed at imposing a physical constraint: all these parameters, whether growth rates, interaction rates, initial population counts, or noise, must be positive by definition. This is not only reasonable prior knowledge but also a necessary condition to guarantee the stability of the numerical solver.
- **Gradient Information and HMC:** Unlike Metropolis-Hastings which relies on random walks, the HMC algorithm samples by simulating Hamiltonian dynamics in a physical system. It utilizes gradient information from the posterior distribution to guide the sampling direction, enabling highly efficient exploration of complex, high-dimensional parameter spaces. In this case, we leverage the `sunode` library, which can automatically and efficiently calculate the gradients of the ODE solution with respect to each parameter. It is precisely this accurate gradient information that allows the NUTS sampler to successfully tackle this challenge.

We can observe that compared to the Metropolis-Hastings MCMC code we implemented in the last lecture, this code takes longer to run. The fundamental reason lies in the use of the computationally more expensive **Hamiltonian Monte Carlo (HMC) algorithm**. In each iteration, Metropolis-Hastings only needs to numerically solve the differential equation (ODE) once to calculate the likelihood; whereas HMC, to achieve more efficient parameter space exploration, requires not only solving the ODE but also solving a more complex adjoint differential equation system via `sunode` to obtain the gradient of the posterior, and using this gradient information to simulate a whole trajectory to generate the next sample.

Why do we still use HMC despite the long computation time? First, HMC offers more comprehensive modeling and more realistic uncertainty. We did not assume the initial values and errors were known. By simultaneously inferring 8 parameters, we obtained a more complete and honest assessment of the uncertainty of the entire system. The MH code in the previous session fixed these values, potentially underestimating the final uncertainty.

Secondly, unmatched sampling efficiency and reliability. This is perhaps the most important advantage. Although MH might give 50,000 samples in 10 minutes, these samples may be highly correlated, effectively worth only a few hundred independent samples. While NUTS takes longer to generate 4,000 samples, they might be equivalent to thousands of independent samples. For complex scientific problems, the “quality” of samples is far more important than “quantity”. The results from HMC are far more reliable than MH, ensuring we have explored the true shape of the posterior distribution.

Actually, students with some data background might wonder: we clearly have faster methods (like `scipy.optimize.curve_fit`) that can give a set of optimal values (a point estimate) in seconds. It tells you “which curve best fits these data points”. But the Bayesian ODE method (ours) gives not just a parameter estimate, but the complete posterior distribution. It tells you: “Based on the data and our physical model, there is a 95% probability that the prey birth rate α lies in the interval $[0.5, 0.8]$, with the most likely value being 0.65”.

This is a world of difference in scientific pursuit. **What we care about is usually not that optimal curve, but the internal laws driving this system, which are the true values of parameters like $\alpha, \beta, \gamma, \delta$.** Understanding these parameters means understanding the ecosystem itself. This is like a doctor diagnosing the cause of an illness, rather than just treating the fever.

Therefore, true science is about Rigorous Uncertainty Quantification, not just data alchemy. In science, engineering, and policy-making, knowing “how certain” we are about a result is crucial. For example, saying “according to the model, there is a 95% probability that sea levels will rise between 10 and 15 cm by 2030” is a statement with rigorous probabilistic guarantees, serving as the cornerstone for climate change policy. Similarly, in drug development, determining the confidence interval for the effective dosage of a new drug is directly related to patient safety. We are willing to spend weeks of computation time to exchange for this reliability.

More importantly, we can squeeze every drop of information out of the data. In many scientific fields, data is extremely precious and scarce. For example:

- Clinical trials: Limited number of patients.
- Paleontology: Scarce fossil samples.
- Space exploration: Limited data transmitted back by probes.

In these scenarios, we cannot use Deep Learning which requires massive amounts of data. But Bayesian methods can combine prior knowledge (our ODE model is a very strong prior) with small amounts of data to yield very robust conclusions. It "squeezes" every drop of information from the data.

HMC utilizes the geometric information of the target distribution (through its gradient) to propose long-distance and efficient moves, significantly improving sampling efficiency compared to diffusion-based methods such as random walks. However, this efficiency comes at a cost: HMC requires calculating the gradient of the logarithmic probability and has more tuning parameters (step size, number of steps), resulting in a longer algorithm runtime.

Reference

- Nonequilibrium Field Theories and Stochastic Dynamics, Prof. Erwin Frey, LMU Munich, Summer Semester 2025
- Non-equilibrium States: Irreversibility and the Implications of Entropy Production — Non-equilibrium Field Theory and Stochastic Dynamics, Interesting Methods PhD
- https://mp.weixin.qq.com/s/6ZPccrywQRUKYNILhP49-Q

References