STAT 37830 Group Project Final Report

Paris Hsu, Yifu Wang, Weixin Wang

December 12, 2025

1 Introduction

Monte Carlo methods are fundamental tools for computing integrals that are difficult or impossible to evaluate analytically [2, 3]. Their performance, however, depends critically on the structure of the target distribution and on the design of the sampling algorithm [1]. In problems with strong nonlinearity or multimodality, standard sampling methods may converge slowly, produce highly biased estimates, or fail entirely. This project investigates these issues by comparing four representative Monte Carlo approaches on a single, intentionally difficult test problem: integration over the 1D double-well potential.

In this report, we consider sampling from one-dimensional target distributions of the general form

$$p(x;\beta) \propto \exp\bigl(-\beta V(x)\bigr), \tag{1}$$

where $V : \mathbb{R} \to \mathbb{R}$ is a potential function and $\beta > 0$ is an inverse temperature parameter that controls the concentration of the distribution around the minima of V.

To study the behavior of the sampling algorithms in multimodal settings, we focus on two representative choices of the potential function: a *double-well (simple)* potential and a *triple-well (complex)* potential, given respectively by

$$V(x) = (x^2-1)^2, \qquad V(x) = \frac{1}{40}(x^2-1)^2(x^2-4)^2(x^2-9)^2. \tag{2}$$

These potentials serve as canonical test examples, as they induce target distributions with multiple well-separated modes and therefore pose nontrivial challenges for sampling algorithms.

For example, for the double-well model, its potential function induces a bimodal target distribution with two symmetric modes located at $x = \pm 1$. As β increases, the distribution becomes increasingly concentrated around these modes, leading to sharper separation and slower mixing.

This structure, as shown in Figure 1, creates a large energy barrier that becomes increasingly sharp as β grows, making the model a canonical example where Markov chain Monte Carlo (MCMC) algorithms struggle with mode-sticking and slow mixing, as we will see later in Section 5.2. At the same time, the symmetry of the distribution guarantees that $E[x] = 0$ for all β, giving us a convenient and exact ground truth for quantifying bias.

Using this problem as a unifying benchmark, we systematically evaluate four sampling methods of increasing sophistication:

1. **Naive Monte Carlo**, which serves as a simple baseline method;

2. **Importance Sampling**, used to investigate how the choice of proposal distribution influences estimator variance and numerical stability;

3. **Metropolis–Hastings**, which is expected to suffer from slow mixing and increased bias at large values of β due to poor exploration across separated modes;

4. **Annealing Strategies**, advanced MCMC techniques designed to mitigate multimodality, including

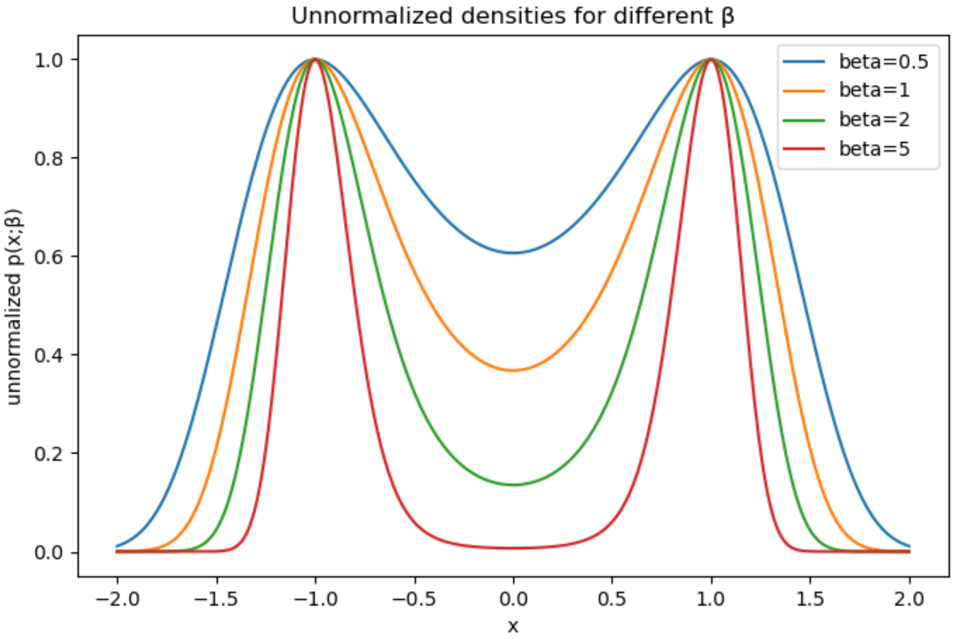

Figure 1: Double-well potential under different β's

(a) Simulated Tempering;

(b) Parallel Tempering.

5. **Gibbs Sampling**, applied to a D-dimensional separable potential to examine whether increased dimensionality alleviates local mode trapping;

6. **Langevin Monte Carlo**, a gradient-based sampling method that leverages local geometric information to promote continuous exploration of the energy landscape.

Our goal is to understand not only how each method performs but why it behaves the way it does. We measure bias, variance, convergence behavior, and computational cost across a range of difficulty settings determined by β. The final product will be a unified, extensible Python toolkit that implements all four methods, along with an analysis module for computing expectations, assessing accuracy, and visualizing sampler behavior.

Through this focused study, we aim to highlight the strengths and limitations of widely used Monte Carlo techniques and to illustrate how problem structure, algorithm design, and tuning choices interact to determine sampling efficiency.

2 Algorithms

2.1 Classical Monte Carlo

Classical Monte Carlo (MC) integration provides a simple baseline for estimating expectations with respect to a target distribution. Given a function $h(x)$ and a distribution with density $f(x)$, s.t. $\int f(x)dx = 1$, the goal is to approximate

$$\mathbb{E}_f[h(X)] = \int h(x)f(x)\,dx. \tag{3}$$

The Classical Monte Carlo method draws N independent samples $X^{(1)}, \ldots, X^{(N)}$ from the target distribution and computes the empirical average

$$\hat{I}_N = \frac{1}{N} \sum_{n=1}^{N} h\Big(X^{(n)}\Big). \tag{4}$$

By the Law of Large Numbers, $\hat{I}_N$ converges to the true expectation as $N \to \infty$, and the estimator has variance $\mathrm{Var}(h(X))/N$.

In our implementation, the user provides two components: a sampling function that returns i.i.d. draws from the target distribution, and a test function $h(x)$ whose expectation is to be estimated. The `estimate` method computes the Monte Carlo estimator $\hat{I}_N$. This method provides a baseline against which we compare the performance of more sophisticated sampling algorithms.

2.2 Importance Sampling

In many problems, drawing samples directly from the target distribution is difficult or computationally expensive. Importance Sampling (IS) provides a way to estimate expectations with respect to a target density $p(x)$ by instead sampling from a more convenient proposal distribution $q(x)$.

Our goal is to compute the expectation

$$\mathbb{E}_p[h(X)] = \int h(x)\, p(x)\, dx, \tag{5}$$

but when sampling from p is impractical, we rewrite the integral by multiplying and dividing by $q(x)$:

$$\mathbb{E}_p[h(X)] = \int h(x)\, \frac{p(x)}{q(x)}\, q(x)\, dx = \mathbb{E}_q[\, h(Y)\, w(Y)\,], \tag{6}$$

where q is a distribution relatively eaiser to sample from, and the importance weight is defined as

$$w(y) = \frac{p(y)}{q(y)}. \tag{7}$$

This identity shows that we can approximate the target expectation using samples $Y^{(1)}, \ldots, Y^{(N)}$ drawn from the proposal distribution:

$$\hat{I}_{\mathrm{IS}} = \frac{\sum_{n=1}^{N} h(Y^{(n)})\, w(Y^{(n)})}{\sum_{n=1}^{N} w(Y^{(n)})}, \tag{8}$$

which is the self-normalized Importance Sampling estimator. This formulation has the advantage that it does not require the normalization constant of $p(x)$.

Moreover, importance sampling can achieve significantly smaller variance than standard Monte Carlo when the proposal distribution is well aligned with the integrand. By reallocating samples toward high-contribution regions, its variance can be bounded in terms of the divergence between the target and proposal, often giving much tighter bounds than the usual $O(1/N)$ rate [6, 7].

In our implementation, the user specifies a proposal sampler and its density $q(x)$, along with the unnormalized target density $p(x)$ and test function $h(x)$. The algorithm generates samples from q, computes the corresponding importance weights, and returns weighted estimates of the desired expectation. A companion routine also provides an approximate variance estimate based on the normalized weights. Importance Sampling serves as both a flexible integration tool and a diagnostic for understanding how the choice of proposal distribution affects estimator variance and stability.

2.3 Metropolis-Hastings (MCMC)

The Metropolis-Hastings (M-H) algorithm is a foundational Markov Chain Monte Carlo (MCMC) method. M-H generates a sequence (or chain) of correlated samples that, in the long run, is guaranteed to settle into and accurately represent the target distribution, $p(x)$.

Algorithm 1 One step of Metropolis–Hastings

```
Input: current state $x$, step size $\sigma$
Propose candidate $x' = x + \mathcal{N}(0, \sigma^2)$
Compute acceptance probability:
    $\alpha = \min\left(1, \frac{p(x';\beta)}{p(x;\beta)}\right)$
Draw $u \sim U[0, 1]$
if $u < \alpha$ then
    $x \leftarrow x'$
else
    $x \leftarrow x$
end if
Output: $x$
```

2.3.1 Detailed Balance and the M-H Mechanism

The core principle ensuring this long-run convergence is detailed balance. A Markov chain satisfies detailed balance with respect to a target distribution $p(x)$ if, for any two states x and x', the probability of being at x and transitioning to x' is equal to the probability of being at x' and transitioning back to x. Mathematically, this is expressed as:

$$p(x)P(x'|x) = p(x')P(x|x') \tag{9}$$

where $P(x'|x)$ is the transition probability from state x to state x'. By satisfying this condition, the chain maintains $p(x)$ as its unique stationary distribution.

At each iteration of the algorithm, the chain evaluates a possible update. It may move to a newly proposed state x', or it may simply remain at its current location x_t. This ”propose–then–accept” mechanism allows the algorithm to explore the state space while automatically correcting for biases introduced by the proposal distribution $q(x'|x)$. The acceptance probability α is carefully constructed to enforce the detailed balance condition.

The corresponding pseudocode is detailed in Algorithm 1.

2.4 Annealing Strategies for sampling

Sampling multi-modal target distributions(e.g. complex wells potential) can be very challenging. Annealing strategies seek to improve sampling by introducing auxiliary tempered distributions that bridge regions of low probability [6].

2.4.1 Simulated Tempering

Simulated tempering relies on the idea that low inverse temperature(high beta) the chain will easily move across the state space, and by the time the augmented chain returns to the temperature of interest $\beta_1 = 1$, the x variable will have moved to a different region of the state space, accelerating the mixing [6].

Here, we provide the detailed pesudocode in Algorithm 2.

The Simulated tempering contains only two Markov chains: one for the state X, the other one for indices of β_k. For each state $X^{(k)}$, it corresponds to β_k. Continually updating β_k, the chain will easily cross the state boundary and effectively sample target distributions.

2.4.2 Parallel Tempering

Parallel tempering, also known as replica exchange Monte Carlo, is studied in the double-well potential, one of two simple free-energy landscapes [4]. By generating hot replicas(low β) and swap them, it can effectively

Algorithm 2 Simulated Tempering

```
Input: Inverse temperatures $\beta_1 = 1 > \cdots > \beta_K$, proposal Markov kernel $q(x, z)$, proposal Markov kernel $r(k, l)$, initialization $(X^{(0)}, k^{(0)})$, sample size $N$.
for $n = 0, 1, \ldots, N-1$ do:
    Draw $\ell^\star \sim r(k^{(n)}, \cdot)$.
    Update:
        $$k^{(n+1)} = \begin{cases} \ell^\star & \text{with probability } a_{k,\ell^\star}, \\ k^{(n)} & \text{with probability } 1 - a_{k,\ell^\star}, \end{cases}$$
    where the acceptance probability is:
        $$a_{k,\ell^\star} := \min\left\{1, \frac{f^{\mathrm{ST}}(X^{(n)}, \ell^\star)}{f^{\mathrm{ST}}(X^{(n)}, k^{(n)})} \frac{r(\ell^\star, k^{(n)})}{r(k^{(n)}, \ell^\star)}\right\}.$$
    Draw $Z^\star \sim q(X^{(n)}, \cdot)$.
    Update:
        $$X^{(n+1)} = \begin{cases} Z^\star & \text{with probability } a_{X,Z^\star}, \\ X^{(n)} & \text{with probability } 1 - a_{X,Z^\star}, \end{cases}$$
    where the acceptance probability is:
        $$a_{X,Z^\star} := \min\left\{1, \frac{f^{\mathrm{ST}}(Z^\star, k^{(n+1)})}{f^{\mathrm{ST}}(X^{(n)}, k^{(n+1)})} \frac{q(Z^\star, X^{(n)})}{q(X^{(n)}, Z^\star)}\right\}.$$
end for
Output: sample $\{(X^{(n)}, k^{(n)})\}_{n=1}^N$.
```

sample multimodal distributions, overcoming the common failure of traditional MCMC methods.

Similarly, we provide the detailed pseudocode in Algorithm 3.

Parallel tempering runs several Metropolis–Hastings chains at different inverse temperatures $\beta_1 < \cdots < \beta_n$. Hot chains explore the state space more freely, while the coldest chain targets the true distribution. At each iteration, every chain performs a Metropolis–Hastings update, followed by a possible swap of states between two *neighboring* temperatures. The swap probability preserves detailed balance across chains, allowing information from the well-mixing hot chains to propagate down to the cold chain. Samples from the coldest chain then approximate the target distribution.

2.5 Gibbs Sampling (Extension to Higher Dimensions)

Gibbs Sampling is a Markov Chain Monte Carlo method that generates samples from a joint distribution $p(\mathbf{x})$ by successively drawing samples from the conditional distribution of each variable. We use this method to extend the analysis to D dimensions, where $\mathbf{x} = (x_1, x_2, \ldots, x_D)$.

We define the D-dimensional separable potential as the sum of D independent 1D double-well potentials:

$$V(\mathbf{x}) = \sum_{i=1}^{D} V(x_i) = \sum_{i=1}^{D} (x_i^2 - 1)^2 \tag{10}$$

The resulting target density is proportional to the D-times product of 1D densities: $p(\mathbf{x}; \beta) \propto \prod_{i=1}^{D} e^{-\beta(x_i^2-1)^2}$.

Algorithm 3 Parallel Tempering

```
**Input:** inverse temperatures $\beta_1 < \beta_2 < \cdots < \beta_n$, initial states $\{X_i^{(0)}\}_{i=1}^n$, sample size $N$.
**for** $t = 0, 1, \dots, N-1$ **do**
    **for** $i = 1, \dots, n$ **do**
        Generate $\widetilde{X}_i^{(t+1)}$ by performing one Metropolis–Hastings step
          with current state $X_i^{(t)}$, proposal kernel $q$, and target density $p_{\beta_i}$.
    **end for**
    For each $\ell, m \in \{1, \dots, n\}$ with $\ell, m$ adjoint, attempt a swap between chains $\ell$ and $m$:
        $\left(X_\ell^{(t+1)}, X_m^{(t+1)}\right) = \begin{cases} \left(\widetilde{X}_m^{(t+1)}, \widetilde{X}_\ell^{(t+1)}\right), & \text{with probability } a_{\ell,m}, \\ \left(\widetilde{X}_\ell^{(t+1)}, \widetilde{X}_m^{(t+1)}\right), & \text{with probability } 1 - a_{\ell,m}, \end{cases}$
    where
        $a_{\ell,m} := \min\left\{1, \frac{p_{\beta_\ell}(\widetilde{X}_m^{(t+1)})\, p_{\beta_m}(\widetilde{X}_\ell^{(t+1)})}{p_{\beta_m}(\widetilde{X}_m^{(t+1)})\, p_{\beta_\ell}(\widetilde{X}_\ell^{(t+1)})}\right\}.$
**end for**
**Output:** the sample $\{X_n^{(t)}\}_{t=1}^N$ from the cold chain $\beta_n$.
```

Because the distribution is separable, the conditional distribution for any component x_i given all other components $\mathbf{x}_{\neg i}$ simplifies to the marginal 1D density:

$$p(x_i|\mathbf{x}_{\neg i}) \propto p(x_i) \propto e^{-\beta(x_i^2-1)^2} \tag{11}$$

Sampling Strategy: Due to this separability, the D-dimensional Gibbs Sampler is implemented by performing D successive Metropolis-Hastings steps, where each step targets the original 1D double-well potential. This setup allows us to rigorously test if the local, 1D mode-sticking failure is replicated or mitigated in higher dimensions.

2.6 Langevin Monte Carlo

We next introduce Langevin Monte Carlo (LMC), a sampling algorithm that shares many conceptual similarities with gradient descent–based optimization methods. Gradient descent seeks to minimize an objective function $V : \mathbb{R}^d \to \mathbb{R}$ by iteratively taking steps in the direction of $-\nabla V$. In contrast, Langevin Monte Carlo aims to sample from a target distribution $f \propto \exp(-V)$ by evolving the state in a stochastic manner that, on average, also follows the direction of $-\nabla V$. Just as gradient descent can rapidly converge to a local minimum of V, Langevin algorithms can efficiently locate and explore high-probability regions (modes) of the target distribution [6].

The algorithm chooses its next step along the inverse gradient descent derction with some random noise, as is shown in Algorithm 4.

3 Package Structure

Structure of the `src` Package The `src` directory contains the core sampling algorithms used throughout the project. At this stage, two methods have been implemented in `src/scripts.py`:

- **ClassicalMonteCarlo**: A direct Monte Carlo estimator that computes expectations of the form

$$\mathbb{E}_f[h(X)] = \int h(x)f(x)\,dx \tag{12}$$

Algorithm 4 Unadjusted Langevin Algorithm (ULA)

```
Input: target density $f(x) \propto \exp(-V(x))$, initial distribution $\pi_0$, step size $\varepsilon > 0$, sample size $N$.
Initial draw: Sample $X^{(0)} \sim \pi_0$.
for $n = 0, 1, \ldots, N-1$ do
    Sample $\xi^{(n)} \sim \mathcal{N}(0, I)$.
    Update
        $$X^{(n+1)} = X^{(n)} - \varepsilon \nabla V\big(X^{(n)}\big) + \sqrt{2\varepsilon}\, \xi^{(n)}.$$
end for
Output: samples $\{X^{(n)}\}_{n=1}^{N}$.
```

Algorithm 5 Gibbs Sampler for Separable Potential $p(\mathbf{x}) \propto \prod_{i=1}^{D} p(x_i)$

```
Input: current state vector $\mathbf{x}$, step size $\sigma$, dimension $D$
Initialize chain $\mathcal{X} = []$
for $iteration = 1$ to $N$ do
    $\mathbf{x}_{\text{current}} \leftarrow \mathbf{x}_{\text{current}}$                ▷ Start the sweep from the last accepted state
    for $i = 1$ to $D$ do
        Sample component $x_i$ from $p(x_i|\mathbf{x}_{\neg i}) \propto p(x_i)$
        $x_{\text{temp}} \leftarrow x_i$
        Propose candidate $x' = x_{\text{temp}} + \mathcal{N}(0, \sigma^2)$
        Compute acceptance probability:
            $$\alpha = \min\left(1, \frac{p(x')}{p(x_{\text{temp}})}\right)$$
        Draw $u \sim U[0,1]$
        if $u < \alpha$ then
            $x_i \leftarrow x'$                ▷ Update the $i$-th component of the current state $\mathbf{x}$
        end if
    end for
    Append $\mathbf{x}_{\text{current}}$ to $\mathcal{X}$
end for
Output: Chain $\mathcal{X}$
```

by drawing i.i.d. samples from the target distribution via a user-provided sampler. The class exposes two methods:

  - `estimate(N)`: returns the empirical mean $\frac{1}{N}\sum_{n=1}^{N} h(X^{(n)})$,

- **ImportanceSampling**: An implementation of self-normalized importance sampling for an unnormalized target density $\tilde{p}(x)$. Given a proposal density $q(x)$, samples from q, and a test function $h(x)$, the estimator computes

$$\hat{I}_{\text{IS}} = \frac{\sum_{n=1}^{N} h(Y^{(n)})\, w(Y^{(n)})}{\sum_{n=1}^{N} w(Y^{(n)})}, \quad w(y) = \frac{\tilde{p}(y)}{q(y)}. \tag{13}$$

  The class provides:

  - `estimate(N)`: returns the self-normalized IS estimator,

- **Metropolis-Hastings (MCMC)**: We based our MCMC implementation structure on a guide provided by Mocz [5]. Located in `src/metropolis.py`, this is an implementation of the symmetric M-H sampler using a Gaussian proposal. This is our primary target for demonstrating sampling failure at high β. The module exposes two key routines:

  - `metropolis_hastings`: Generates the MCMC chain of correlated samples. This function is used to demonstrate the issue of high bias at large β values.
  - `tune_step_size`: Provides an automated pre-tuning routine to optimize the proposal's σ for a target acceptance rate (e.g., 40%), maximizing mixing efficiency.

- **Simulated tempering**: Located in `src/simulated_tempering.py`, this is an implementation of Simulated tempering This module consists of four functions:

  - `V_1, V_2 and log_p`: define the potential function

$$V_1(x) = (x^2 - 1)^2, \qquad V_2(x) = \frac{1}{40}(x^2 - 1)^2(x^2 - 4)^2(x^2 - 9)^2 \tag{14}$$

    and the log version of target density function

$$\log(P(x; \beta)) \propto -\beta V(x) \tag{15}$$

  - `generate_betas`: generates an array $\{\beta_{\max} = \beta_1, \beta_2, \dots, \beta_n = \beta_{\min}\}$ by logarithmical spacing, assuming that $\beta_i = 1/T_i$, where T_i is one of the temperatures.
  - `simulated_tempering`: apply Simulated tempering to $\{\beta_1, \beta_2, \dots, \beta_n\}$ and generate samples X_i and corresponding indices k_i of β_k.

- **Parallel tempering**: Located in `src/parallel_tempering.py`, this is an implementation of Parallel Tempering for *Metropolis-Hastings* algorithm (Note that this M-H may not be the same as previous one). This module consists of four functions:

  - `V_1, V_2 and log_p`: define the potential function

$$V_1(x) = (x^2 - 1)^2, \qquad V_2(x) = \frac{1}{40}(x^2 - 1)^2(x^2 - 4)^2(x^2 - 9)^2 \tag{16}$$

    and the log version of target density function

$$\log(P(x; \beta)) \propto -\beta V(x) \tag{17}$$

  - `generate_betas`: generates an array $\{\beta_{\max} = \beta_1, \beta_2, \dots, \beta_n = \beta_{\min}\}$ by logarithmical spacing, assuming that $\beta_i = 1/T_i$, where T_i is one of the temperatures.
  - `parallel_tempering`: apply Parallel tempering to $\{\beta_1, \beta_2, \dots, \beta_n\}$ and generate samples X_i.

- **Gibbs Sampling (MCMC Extension)**: To test the mode-sticking phenomenon in higher dimensions, we implemented a Gibbs Sampler for a separable D-dimensional double-well potential, $V(\mathbf{x}) = \sum_{i=1}^{D}(x_i^2 - 1)^2$. The code for this routine is located in `src/gibbs.py` and relies on an internal Metropolis-Hastings step defined in `src/metropolis.py`.

  - `metropolis_step_1d`: A specialized routine in `src/metropolis.py` that executes a single 1D M-H step. This function is called repeatedly by the Gibbs sampler.
  - `gibbs_sampler`: Executes the D-dimensional MCMC chain by performing D successive updates of `metropolis_step_1d` per iteration. This function is used to prove that the localized mixing failure persists regardless of the dimensionality D.

- **Langevin Monte Carlo**: We implement the *Unadjusted Langevin Algorithm (ULA)* for sampling from a target density $f(x) \propto \exp(-V(x))$ in $\mathbb{R}^d$, where the drift term is given by the gradient of the potential $\nabla V(x)$.

- The implementation is encapsulated in a Python class `UnadjustedLangevin`, which takes as inputs the gradient function `gradV`, a step size $\varepsilon > 0$, and a sampler for the initial distribution π_0.
- Samples are generated via the Euler–Maruyama discretization

$$X^{(n+1)} = X^{(n)} - \varepsilon \nabla V(X^{(n)}) + \sqrt{2\varepsilon}\, \xi^{(n)}, \qquad \xi^{(n)} \sim \mathcal{N}(0, I), \tag{18}$$

  producing a Markov chain whose stationary distribution approximates f.
- In addition to chain generation, the class provides an `estimate` method for Monte Carlo estimation of expectations $\mathbb{E}_f[h(X)]$, supporting burn-in and thinning to reduce initialization bias and sample correlation.

4 Verification and Testing

To ensure the reliability of our simulations, we used the `pytest` framework to write thorough unit tests, primarily focusing on proving that our samplers behave exactly as predicted by theory.

Classical Monte Carlo Tests (`test_for_classicalMC.py`) These tests ensure that the basic Monte Carlo estimator behaves consistently with theoretical expectations:

- **Mean and Second Moment Tests:** Using $X \sim \mathcal{N}(0,1)$, we verify that the empirical estimates of $\mathbb{E}[X]$ and $\mathbb{E}[X^2]$ converge to their known analytical values. This confirms correct implementation of the sampler and averaging routine.
- **Variance Consistency Test:** We compute $\mathbb{V}[X]$ both directly and via the identity $\mathbb{V}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$ and check that the estimates agree within sampling noise, ensuring internal numerical consistency.

Importance Sampling Tests (`test_for_IS.py`) These tests validate that the self-normalized importance sampling estimator is implemented correctly for an unnormalized double-well target density. We test the self-normalized Importance Sampling estimator on the double-well target density $p(x) \propto \exp(-\beta(x^2-1)^2)$ with $\beta = 1$. A Gaussian proposal $q = \mathcal{N}(0, 2^2)$ is used to draw samples and compute importance weights $w = p(x)/q(x)$.

- The test estimates the moments $\mathbb{E}[X]$, $\mathbb{E}[X^2]$, and $\mathbb{E}[X^3]$ to verify that the weighting and normalization are implemented correctly.
- A separate plug-in estimator for $\mathbb{V}[X]$ confirms the numerical stability of the variance computation under the IS framework.

Analysis Module Tests (`test_analysis.py`) These tests verify that our core measurement tools are correct:

- **Moment Calculation:** We verified that `compute_expectation` accurately calculates moments ($E[x^n]$) for both **unweighted** MCMC samples (simple average) and **weighted** Importance Samples (self-normalized mean). This confirms the function works for all three current samplers.
- **Metric Calculation:** We tested `calculate_metrics` with known inputs to confirm its ability to correctly compute the experimental **bias** and **sample variance** of an estimator across multiple runs.

Metropolis-Hastings Tests (`test_metropolis.py`) These tests focus on experimentally proving the mode-sticking phenomenon in finite-length MCMC simulations.

- **Smoke Test:** We verified that the `metropolis_hastings` function executes cleanly and returns the correct number of samples after burn-in.
- **Trap Verification Test:** This is the key diagnostic test. We run the sampler with parameters known to cause failure ($\beta = 20.0$, small `step_size`, starting at $x = 1.0$) and assert two critical conditions:
  1. **Sanity Check ($E[x^2]$):** We assert that the estimate for $E[x^2]$ is near the true value (≈ 1.0). This proves the sampler is correctly finding the location of the peak.
  2. **Failure Check ($E[x]$):** We assert that the estimate for $\mathbb{E}[x]$ is near 1.0 (and not 0.0). The result from this finite simulation demonstrates that the chain is trapped in one mode, serving as empirical proof of the high bias resulting from incomplete exploration of the state space (due to slow mixing), which we aim to overcome.
- **Tuner Plausibility Test:** We verified that the `tune_step_size` function successfully converges and returns a positive, plausible step size within the optimal 40-50% acceptance rate range.

Problem and Analysis Utility Tests (`test_problem_and_analysis.py`) These tests verify the correctness of the ground truth moments, potential functions, and density calculation utilities used by all MCMC experiments. This ensures the benchmarks used to validate the samplers are accurate.

- **Potential Symmetry Tests:** We verified that both the simple double-well potential and the complex multi-well potential are mathematically symmetric, ensuring that all odd true moments ($\mathbb{E}[x], \mathbb{E}[x^3]$, etc.) are correctly zero.
- **Potential Value Tests:** We confirmed that the complex potential (`complex_potential`) returns the expected values at known critical points (minima at $x = \pm 1, \pm 2, \pm 3$ and barrier at $x = 0.0$).
- **Generalized Moment Test:** We verified that the generalized `get_true_even_moment` function correctly calculates the ground truth via numerical quadrature, showing backward compatibility with known values (e.g., $\mathbb{E}[x^2] \approx 0.8327$ for $\beta = 1.0$).
- **Plotting Smoke Test:** We verified that the generalized `plot_histogram` function successfully generates and plots the true density curve for both the simple and complex potentials without crashing.

Simulated Tempering Tests (Test_ST.py) It includes the functions that check whether the output of previous functions is what we want:

- `test_simulated_tempering_V1`: apply ks-test to check whether `simulated_tempering` outputs an array with correct distribution, applying the potential function V_1.

Parallel Tempering Tests (Test_PT.py) It includes the functions that check whether the output of previous functions is what we want:

- `test_V1`: check whether `V_1` output correctly.
- `test_V2`: check whether `V_2` output correctly.
- `test_log_p`: check whether `log_p` output correctly.
- `test_generate_betas`: check `generate_betas` whether generate an array with correct length(number of β_i), order and spacing.
- `test_parallel_tempering`: check `parallel_tempering` whether output an array with correct length (number of state t) and under correct distribution.

Gibbs Sampling Tests (`test_gibbs.py`) These tests verify the robust functionality of the D-dimensional Gibbs sampler and confirm that the mode-sticking failure persists when moving to higher dimensions.

- **M-H Component Tests:** We verified that the refactored M-H sampler and its tuner operate correctly across different potential densities (e.g., complex potential) and that the internal `metropolis_step_1d` routine handles edge cases.
- **Smoke and Shape Test:** The `gibbs_sampler` is asserted to run cleanly for low β ($\beta = 1.0$) and return a finite matrix with the correct dimensions ($N \times D$).
- **High-Dimensional Trap Verification (Key Diagnostic):** The sampler is run under high-failure conditions ($\beta = 20.0$, small σ). The analysis of the first component (x_1) asserts:
  1. **Sanity Check ($E[x_1^2]$):** The estimate (≈ 0.987) confirms the chain is correctly locating the local mode.
  2. **Failure Check ($E[x_1]$):** The estimate (≈ 1.0) proves the chain is stuck in a single mode, demonstrating that increasing D does not solve the localized mixing failure of component-wise samplers.

Unadjusted Langevin Test

- We test the Unadjusted Langevin Algorithm on a one-dimensional *tri-well* target distribution with potential $V(x) = \frac{1}{40}(x^2 - 1)(x^2 - 4)(x^2 - 9)$, whose gradient is implemented analytically.
- Using a step size $\varepsilon = 0.01$ and inverse temperature $\beta = 1$, we generate long ULA trajectories and discard an initial burn-in of 2×10^4 steps to reduce initialization bias.
- Monte Carlo estimators are constructed for the first three moments $\mathbb{E}[X]$, $\mathbb{E}[X^2]$, and $\mathbb{E}[X^3]$, as well as the variance, by evaluating appropriate test functions along the Markov chain.
- The resulting moment estimates provide a quantitative validation of the ULA implementation and demonstrate its ability to explore a multimodal target distribution.

5 Results

5.1 Classical Monte Carlo and Importance Sampling

The provided script implements two separate numerical experiments:

- A `ClassicalMonteCarlo` estimator for expectations under the standard normal distribution $N(0, 1)$;

  For the classical Monte Carlo component, the code samples directly from $N(0, 1)$ and computes empirical estimates of the mean and variance for increasing sample sizes N.
- A `ImportanceSampling` estimator for expectations under the unnormalized double-well target density

  $$p(x) \propto e^{-\beta(x^2-1)^2}. \tag{19}$$

  For the importance sampling component, the script draws samples from a Gaussian proposal distribution $q(x) = N(0, 2^2)$ and applies normalized importance weights to estimate the mean and variance of the double-well target distribution.

The experiment sweeps over a logarithmic grid of sample sizes $N \in [10^2, 10^5]$, records the corresponding estimates, and visualizes the behavior of these estimates as N increases, which is shown in Figure 2.

Taken together, these four panels illustrate how both classical Monte Carlo and importance sampling estimators improve with larger sample sizes, each for its respective target distribution.

As additional sampling modules are introduced, corresponding tests will be added to ensure correctness and compatibility across the package.

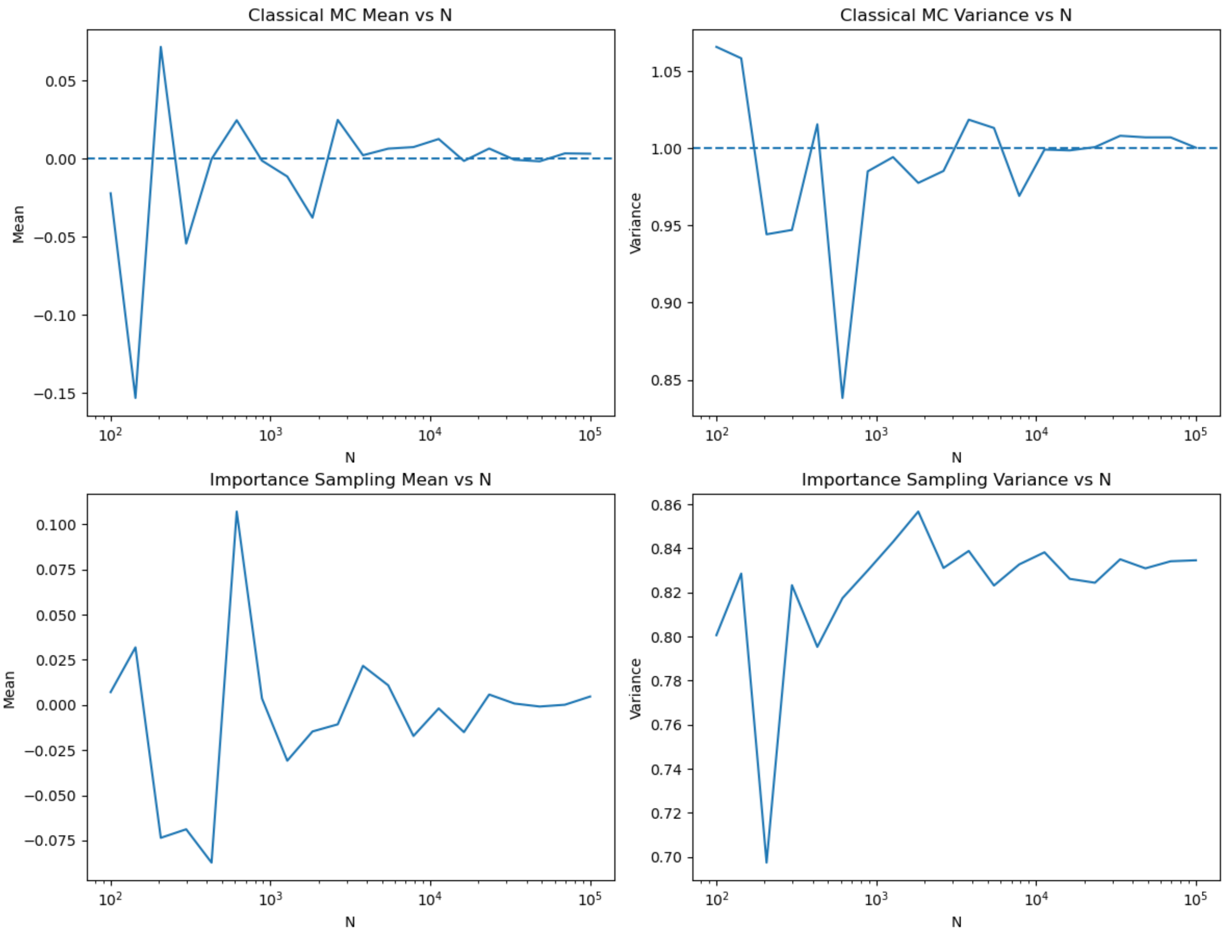

Figure 2: empirical mean and variance of classical Monte Carlo (top row) and importance sampling (bottom row) estimators w.r.t. the sample size.

5.2 Metropolis-Hastings Sampling

In this numerical experiment, we sample from the unnormalized *double-well target density*

$$p(x) \propto e^{-\beta(x^2-1)^2}. \tag{20}$$

This distribution is bimodal with two local maxima (wells) centered at $x \approx \pm 1$, with the parameter β controlling the sharpness of the modes. The performance of the M-H sampler is highly dependent on its ability to traverse the energy barrier separating these modes, a phenomenon known as *mixing*.

As discussed in Section 1, the Metropolis–Hastings chain can become "trapped" when the energy barrier is sharp, leading to slow mixing or even stagnation. Below we illustrate two representative cases: In the first, when $\beta = 1$ is moderate, the chain can still transition between modes; In the second, for large $\beta = 20$, the chain becomes effectively stuck and fails to explore the state space.

5.2.1 Metropolis-Hastings Performance at Moderate β ($\beta = 1$)

For a lower value of $\beta = 1.0$ and an auto-tuned step size, the M-H chain demonstrates **excellent mixing** (Figure 3).

- **Trace:** The chain (top plot) successfully explores both modes, frequently jumping across the barrier between $x \approx -1$ and $x \approx 1$.
- **Histogram:** The MCMC samples (blue histogram, bottom plot) closely match the True Density (red curve), accurately capturing the symmetric bimodal structure.
- **Moments:** The estimates for $\mathbb{E}[x]$ and $\mathbb{E}[x^3]$ correctly converge to zero, confirming the chain has adequately sampled both symmetric modes. All moment estimates are predicted correctly (Table 1).

  Naive SEM (±)[1]

Table 1: Experiment Results for $\beta = 1.0$

Moment	Estimate	Naive SEM (±)[1]	True Value
$\mathbb{E}[x]$	0.0096	0.004096	0.0000
$\mathbb{E}[x^2]$	0.8389	0.002789	0.8327
$\mathbb{E}[x^3]$	0.0121	0.005861	0.0000
$\mathbb{E}[x^4]$	1.0925	0.006109	1.0827

5.2.2 Metropolis-Hastings Performance at High β ($\beta = 20$)

Table 2 summarizes the Monte Carlo estimates of several moments when the inverse temperature is increased to $\beta = 20$. Consistent with the discussion in Section 1, the Metropolis–Hastings chain exhibits *poor mixing* at this large β, becoming trapped in a single mode and failing to explore the full state space. As a result, some moment estimates are badly biased.

Table 2: Moment Estimates of M-H Sampling for $\beta = 20.0$

Moment	Estimate	Naive SEM (±)[1]	True Value	Result
$\mathbb{E}[x]$	0.9899	0.000362	0.0000	Stuck!
$\mathbb{E}[x^2]$	0.9865	0.000711	0.9870	Correct
$\mathbb{E}[x^3]$	0.9893	0.001059	0.0000	Stuck!
$\mathbb{E}[x^4]$	0.9984	0.001416	0.9995	Correct

When the step size is restricted to a small value ($\delta = 0.1$), the chain becomes stuck in the positive mode for the entire duration of the simulation, failing to cross the high-energy barrier separating the two modes, as shown in Figure 6.

- **Trace:** Starting at $x_0 = 1.0$, the chain (top plot) becomes *stuck* in the positive mode for the entire duration of the simulation, failing to cross the now-higher energy barrier to reach the negative mode at $x \approx -1$.
- **Histogram:** The resulting sample histogram (bottom plot) is severely *biased*, only covering the positive mode and failing to reproduce the symmetric, bimodal true density.
- **Moments:** The odd moments, $\mathbb{E}[x]$ and $\mathbb{E}[x^3]$, are incorrectly estimated to be non-zero (≈ 1.0), which is classified as “stuck” because the true value is zero due to symmetry. This highlights the risk of *inaccurate estimates* due to poor mixing and the high-energy barrier effect.

[1] The Standard Error of the Mean (SEM) estimates the statistical precision of an estimator. The Naive SEM assumes independent samples (i.i.d.). This assumption is violated by correlated MCMC samples, meaning the Naive SEM is a lower bound on the true error. However, it is used here to show that the simulation-level bias (the high beta result) significantly dominates the statistical error.

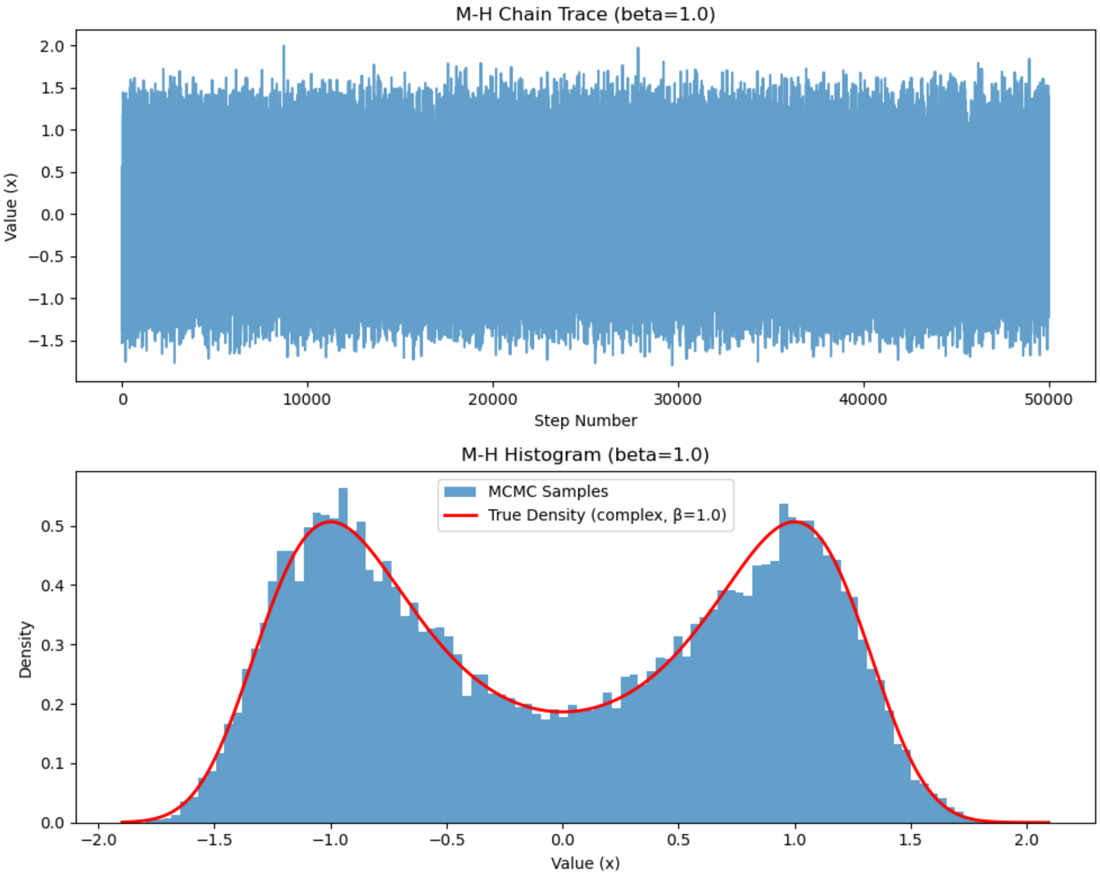

Figure 3: M-H chain trace (top) and histogram (bottom) for the working chain with $\beta = 1$. The chain mixes well and accurately reproduces the bimodal target distribution.

5.2.3 Extension: M-H on the Complex Potential

To assess the severity of slow mixing, we apply the Metropolis-Hastings sampler to a more complex, multi-modal potential: $V_{\text{complex}}(x) = (x^2-1)(x^2-4)(x^2-9)/40$. This potential exhibits multiple symmetric wells, further challenging the chain's ability to mix.

Complex Potential Performance at Moderate β ($\beta = 1.0$)

For a moderate inverse temperature $\beta = 1.0$, the auto-tuned chain successfully navigates the multiple barriers (Figure 5). The odd moments are correctly estimated near zero, confirming the chain successfully explored all modes (Table 3).

- **Trace:** The chain (top plot) successfully explores the multi-well landscape, exhibiting frequent large jumps necessary to traverse the multiple barriers and visit all modes.
- **Histogram:** The MCMC samples (blue histogram, bottom plot) closely match the True Density (red curve), accurately capturing the complex, multi-modal structure.
- **Moments:** The estimates for odd moments ($\mathbb{E}[x]$ and $\mathbb{E}[x^3]$) correctly converge to zero (Table 3), demonstrating excellent mixing across the entire state space.

Complex Potential Performance at High β ($\beta = 20.0$)

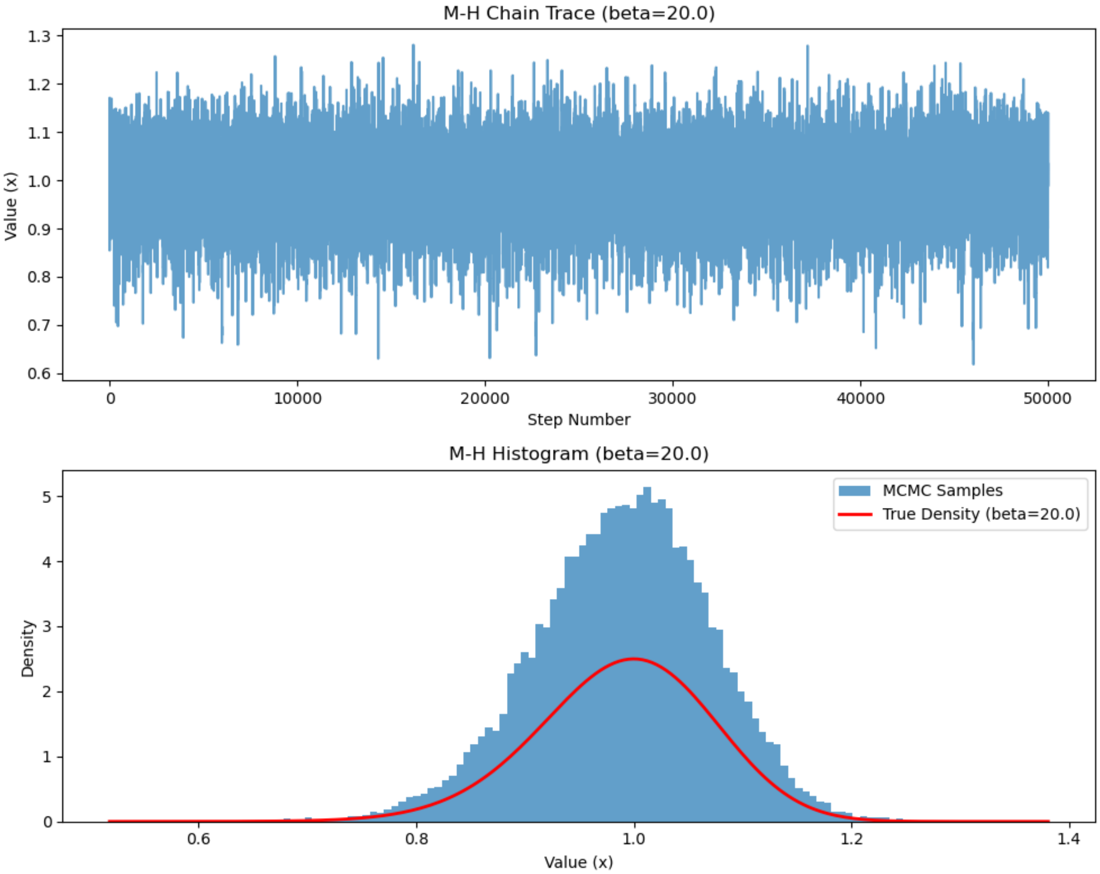

Figure 4: M-H chain trace (top) and histogram (bottom) for the stuck chain with $\beta = 20$. The chain remains confined to a single mode, resulting in a biased sample and incorrect estimation of odd moments.

When β is increased to 20.0 and the chain is initialized at an extreme mode ($x_0 = 3.0$) with a small step size ($\delta = 0.05$), the sampler exhibits extreme mode-sticking.

- **Trace:** Starting at $x_0 = 3.0$, the chain is permanently confined to a narrow region around one positive mode (around $x \approx 2.6$), failing to explore the vast state space.
- **Histogram:** The resulting sample histogram is severely biased, only covering a fraction of the distribution and failing to represent the symmetric density.
- **Moments:** The odd moments ($\mathbb{E}[x]$ and $\mathbb{E}[x^3]$) are significantly non-zero (Table 4), while the even moments are also badly estimated ($\mathbb{E}[x^2] \approx 6.96$ vs. True 3.49). This multiple-moment failure confirms that the mode-sticking phenomenon remains robust even in complex multi-modal energy landscapes.

5.3 Annealing Strategies for sampling

Firstly, look at plot 7, illustrating how good these samples that two tempering methods generated estimate the 1st, 2nd, and 3rd moment of our target density function:

- Comparing left column with right column, it is obviously that the fluctuation range on the left is smaller, indicating that the overall error of Parallel tempering is smaller than that of simulated tempering.

Table 3: Moment Estimates of M-H Sampling for Complex Potential ($\beta = 1.0$)

Moment	**Estimate**	**Naive SEM (±)**[1]	**True Value**	**Result**
$\mathbb{E}[x]$	-0.0013	0.008193	0.0000	Correct
$\mathbb{E}[x^2]$	3.3562	0.013971	3.3468	Correct
$\mathbb{E}[x^3]$	0.0116	0.054428	0.0000	Correct
$\mathbb{E}[x^4]$	21.0237	0.115250	20.9721	Correct

Table 4: Moment Estimates of M-H Sampling for Complex Potential ($\beta = 20.0$)

Moment	**Estimate**	**Naive SEM (±)**[1]	**True Value**	**Result**
$\mathbb{E}[x]$	2.6373	0.000327	0.0000	Wrong (Stuck!)
$\mathbb{E}[x^2]$	6.9607	0.001721	3.4894	Wrong
$\mathbb{E}[x^3]$	18.3857	0.006794	0.0000	Wrong (Stuck!)
$\mathbb{E}[x^4]$	48.5996	0.023868	24.2789	Wrong

- From the lines in the plot, we notice that the purple line, in both Parallel tempering and Simulated tempering, is the most fluctuated one, indicating, two methods cannot deal with the case that potential function $V(x)$ has complex modal-like and high β increases its difficulty.
- we noticed that when estimating 2nd moment by Simulated tempering, the blue line is not continuously. We guess it is due to random initial states $X^{(0)}$ which sometimes were trapped in local minimums.

In order to verify their performance, we need to check the log-log plot of error 8:

- Through comparison, we found that parallel tempering performed better, especially when estimating the 1st moment: all the lines, despite their fluctuations, still exhibit a downward trend (even if it's slow). The other two images in the left column also show a similar trend.
- Overall, when $V(x) = (x^2 - 1)^2$ and $\beta = 1, 20$, both models have relatively small prediction errors for the results: the red line and blue line are located at the bottom of all the charts and both have a negative trend.
- For Simulated tempering, it has difficulty handling high-slope cases: the purple line, denoting $V(x) = \frac{1}{40}(x^2 - 1)^2(x^2 - 4)^2(x^2 - 9)^2$ and $\beta = 20$, does not show a downward trend, but only fluctuated significantly. This usually means that temperature ladders cannot cross the barriers to reach the other "peak".

5.4 Gibbs Sampling: High-Dimensional Failure

The Gibbs Sampling extension applies a component-wise update strategy to the D-dimensional separable double-well potential. We analyze the trajectory and moments of the first component, x_1, as it is representative of the entire system due to symmetry.

5.4.1 Gibbs Sampler Performance at Moderate β ($\beta = 1.0$)

For $\beta = 1.0$ and $D = 10$, the Gibbs sampler successfully explores the state space.

- **Mixing:** The tuner finds an optimal step size (≈ 1.78) which allows each component to frequently transition between its ±1 modes.

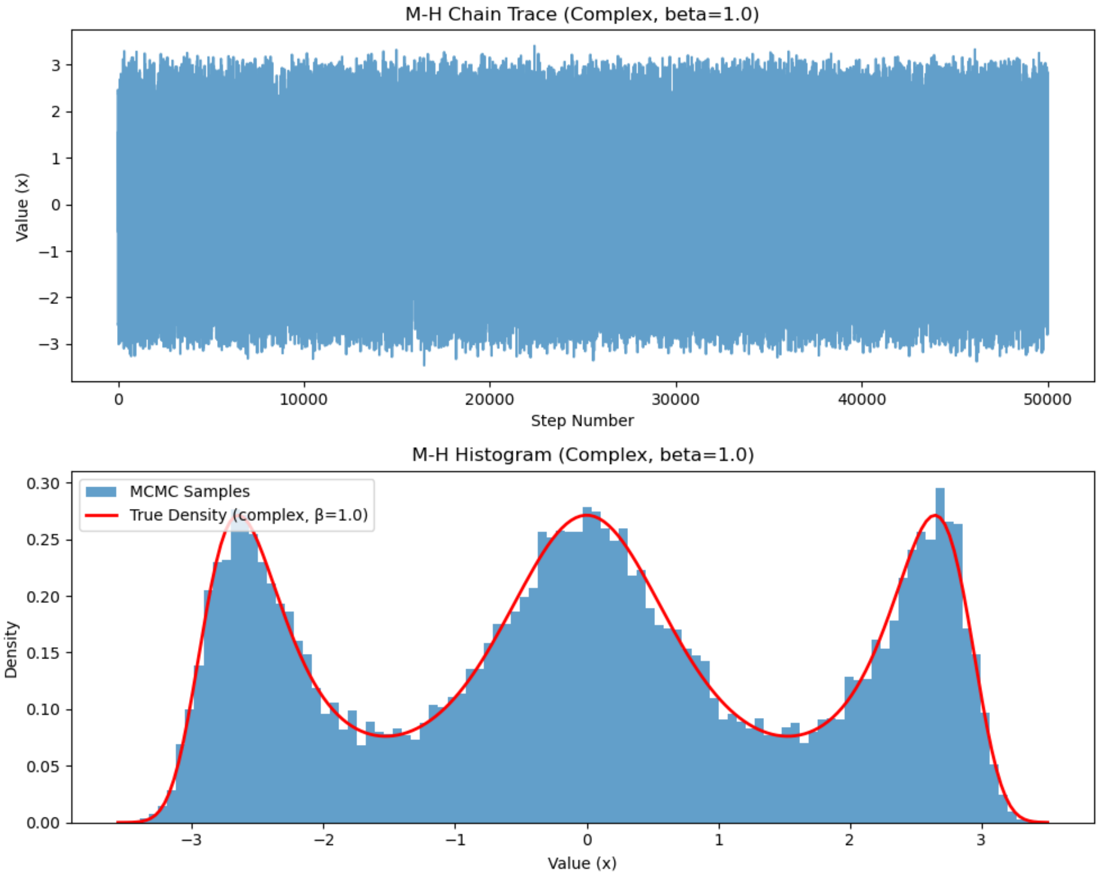

Figure 5: M-H chain trace (top) and histogram (bottom) for the complex potential with $\beta = 1.0$. The chain successfully mixes across the multiple symmetric modes.

- **Moments:** The odd moments ($\mathbb{E}[x_1], \mathbb{E}[x_1^3]$) are correctly estimated near zero (Table 5, Table 6, and Table 7), confirming that when the 1D barrier is low, Gibbs sampling achieves correct convergence.

5.4.2 Gibbs Sampler Performance at High β ($\beta = 20.0$)

When the difficulty parameter is increased to $\beta = 20.0$, the Gibbs sampler exhibits the same failure mode as the 1D M-H sampler, regardless of dimensionality ($D = 5$ or $D = 20$).

- **Failure Mode:** In all high-β cases, $\mathbb{E}[x_1]$ is strongly biased toward the starting value (≈ 0.99).
- **Robustness of Slow Mixing:** The persistence of the $\mathbb{E}[x_1]$ bias proves that increasing the dimensionality D does not alleviate the problem of slow mixing for this separable potential. Since the Gibbs update is localized to a single dimension, the exponentially high energy barrier in that dimension prevents the chain from transitioning to the opposite mode.

This result provides a definitive argument for the necessity of Parallel Tempering, which employs global moves (swapping states between replicas) to overcome the universally localized mixing failures seen across all MCMC methods tested.

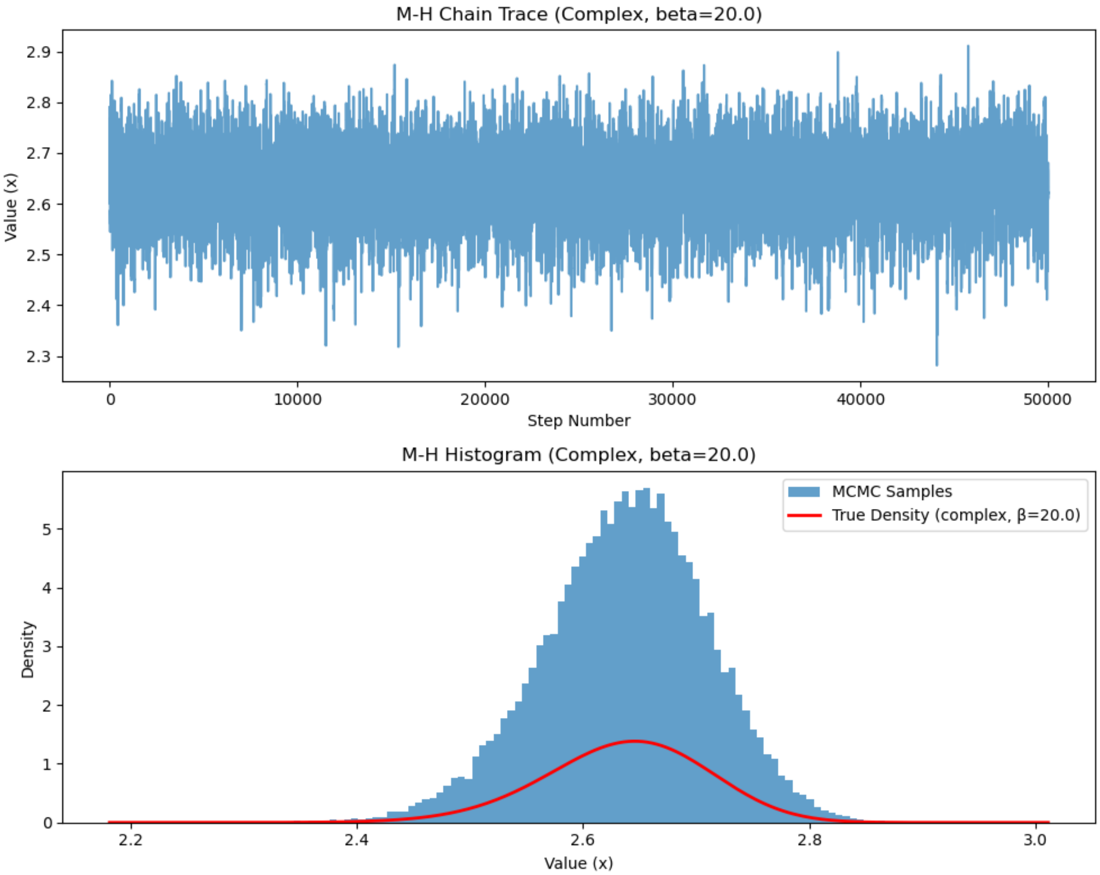

Figure 6: M-H chain trace (top) and histogram (bottom) for the complex potential with $\beta = 20.0$. The chain's failure to transition between modes leads to significant bias, confirming the robustness of slow mixing when the energy barrier is exponentially large.

Note on Potential Choice for Gibbs Sampling: We used the separable double-well potential ($V(\mathbf{x}) = \sum V_{\text{simple}}(x_i)$) for this analysis. While a separable extension of the complex potential was also possible, it would have yielded the same conclusion: the Gibbs sampler fails due to localized 1D jump failures because the core mixing problem is dictated only by the barrier height β. The simple separable model is sufficient to conclusively demonstrate that component-wise MCMC methods universally fail due to localized mixing problems, regardless of the number of dimensions.

The trace estimate and the histogram of the samplings are presented in Figures 9, 10, and 11, while the visualizations of the convergence speed are presented in the log-log plots (Figures 12 and 13), one for each moment.

5.4.3 Low Barrier, $\beta = 1.0$

The three working chains (M-H Simple, M-H Complex, Gibbs D=10) confirm the theoretical foundation of MCMC sampling.

- **Visual Confirmation:** The error lines for all four moments visually track the theoretical $C \cdot n^{-1/2}$ line .
- **Interpretation:** This demonstrates the expected convergence rate. When the energy barrier is low,

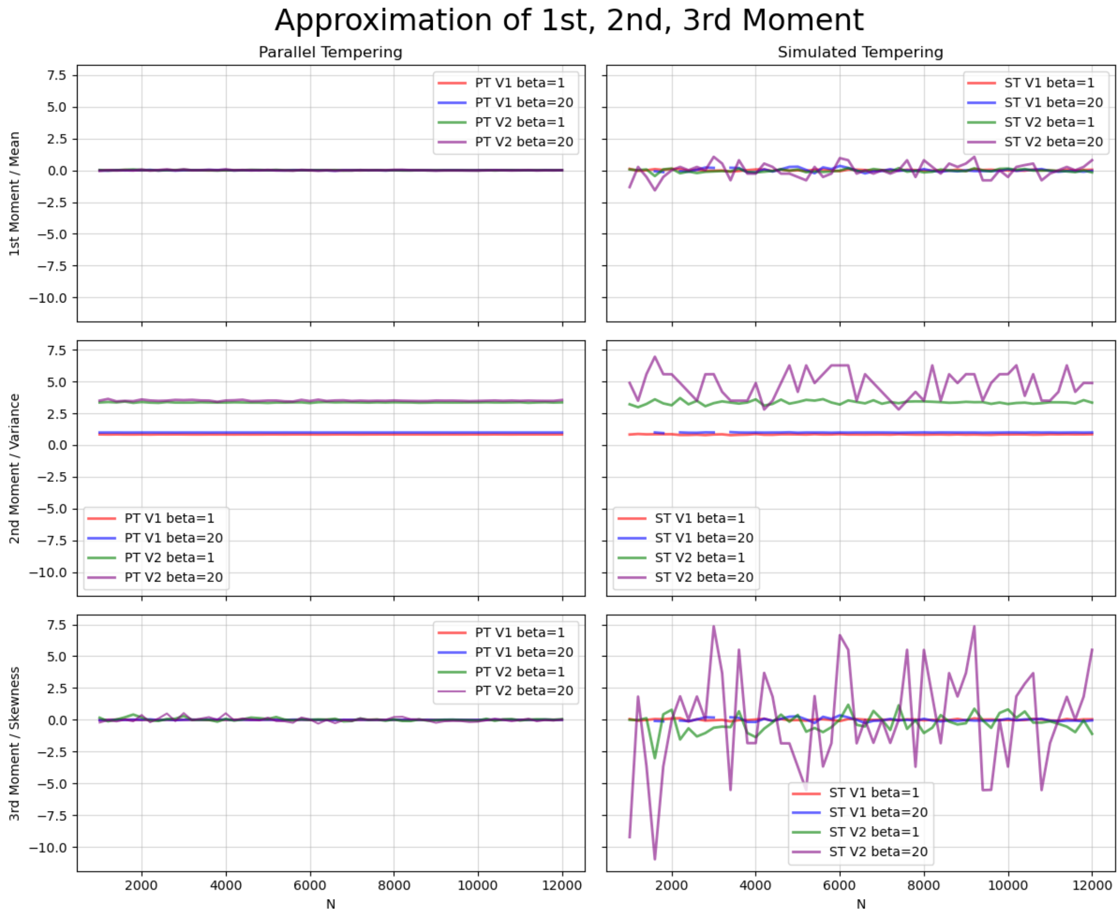

Figure 7: Estimation of 1st, 2nd, and 3rd Moment with target density function $p(x;\beta) = e^{-\beta V(x)}$ by the samples generated by Parallel tempering and Simulated tempering. Here, V1 denotes normal potential function $V(x) = (x^2-1)^2$ and V2 denotes complex potential function $V(x) = \frac{1}{40}(x^2-1)^2(x^2-4)^2(x^2-9)^2$.

the samplers achieve global exploration, and the remaining statistical uncertainty decays efficiently with the square root of the number of samples (N).

5.4.4 High Barrier, $\beta = 20.0$

The three stuck chains (M-H Simple, M-H Complex, Gibbs D=20) definitively expose the localized mixing failure.

- **Odd Moments ($\mathbb{E}[x], \mathbb{E}[x^3]$):** These error lines are the primary diagnostic of failure. After a sharp initial drop, the error lines for the stuck chains become horizontal and flat (scaling as $O(1)$). This constant, high-error floor proves that the sampler is permanently biased, as increasing N provides no improvement in the accuracy of the estimator.

- **Even Moments ($\mathbb{E}[x^2], \mathbb{E}[x^4]$):** The error lines for the even moments successfully decay along the $n^{-1/2}$ slope, even for the stuck chains. This happens because the samplers are locally efficient within the single mode they are trapped in, and $\mathbb{E}[x^n]$ is symmetric, meaning the estimated value is locally correct.

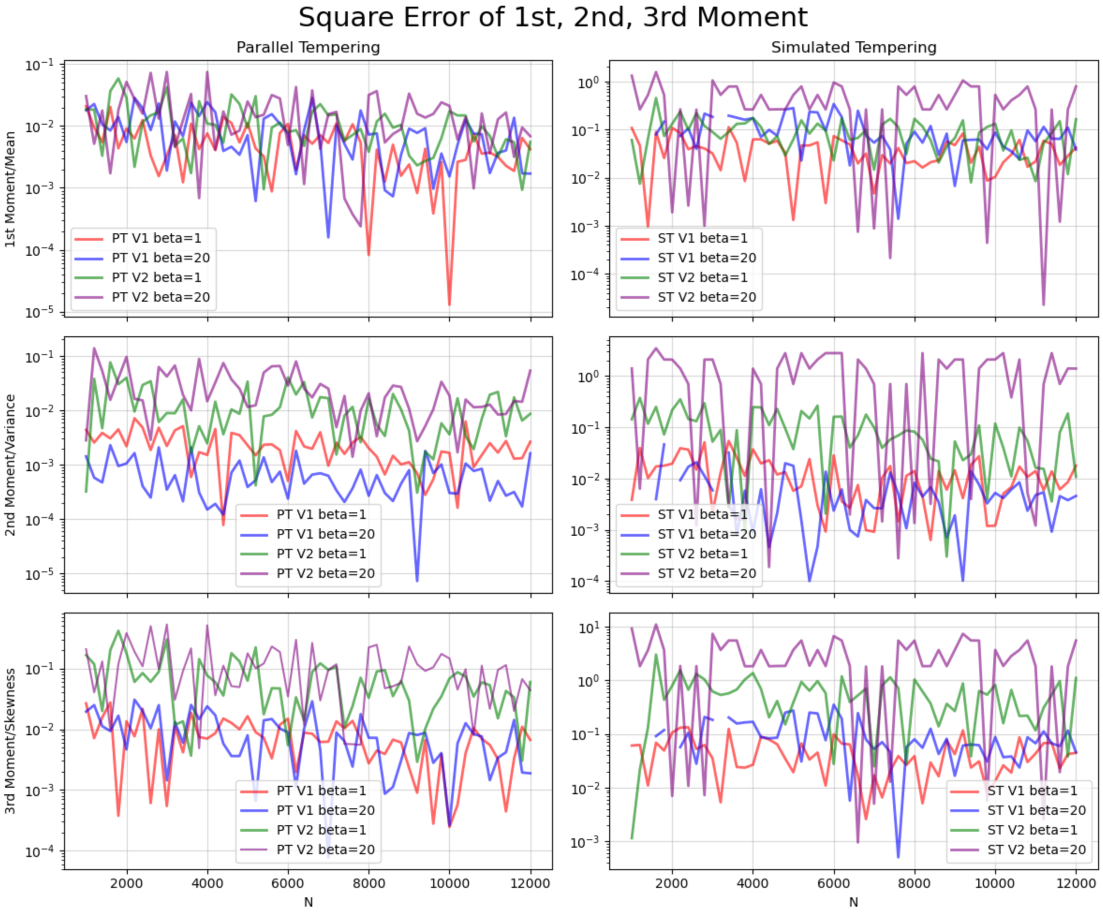

Figure 8: Error in estimating for Parallel tempering and Simulated tempering.

5.5 Unadjusted Langevin

We report a log–log plot of the root mean square error (RMSE) as a function of the number of samples N for the bi-modal and tri-modal potentials with inverse temperatures $\beta = 1$ and $\beta = 20$; see Figure 14.

Overall, the results indicate that the Unadjusted Langevin Algorithm exhibits favorable error decay behavior for moderate β. For $\beta = 1$, the empirical convergence trend is broadly consistent with the expected $\mathcal{O}(N^{-1/2})$ Monte Carlo rate, and the error decreases steadily once the chain has sufficiently explored all modes of the target distribution.

Compared with sampling strategies based on discrete proposal and acceptance mechanisms—such as pick-and-choose or rejection-based methods—Langevin dynamics enjoys a clear advantage in this multimodal setting. As a gradient-based method, ULA is able to "move" continuously along the energy landscape and is therefore less sensitive to unfavorable initializations. In contrast, methods relying on local proposal selection may become trapped when probability mass is highly concentrated in narrow regions: poorly chosen initial states can lead to low acceptance probabilities and long periods of stagnation.

This behavior becomes particularly evident when contrasted with, for example, the annealing-based strategies as in Figure 8. While annealing can mitigate metastability, it introduces additional tuning complexity. In the present setting, ULA with a fixed $\beta = 1$ already achieves robust performance, highlighting the practical effectiveness of gradient-driven sampling for moderately multimodal targets.

Table 5: Moment Estimates of Gibbs Sampling for $D = 10$ ($\beta = 1.0$)

Moment	**Estimate**	**Naive SEM** $(\pm)^1$	**True Value**	**Result**
$\mathbb{E}[x_1]$	0.0014	0.004079	0.0000	Correct
$\mathbb{E}[x_1^2]$	0.8319	0.002796	0.8327	Correct
$\mathbb{E}[x_1^3]$	0.0125	0.005858	0.0000	Correct
$\mathbb{E}[x_1^4]$	1.0829	0.006210	1.0827	Correct

Table 6: Moment Estimates of Gibbs Sampling for $D = 5$ ($\beta = 20.0$)

Moment	**Estimate**	**Naive SEM** $(\pm)^1$	**True Value**	**Result**
$\mathbb{E}[x_1]$	0.9902	0.000361	0.0000	Wrong (Stuck!)
$\mathbb{E}[x_1^2]$	0.9870	0.000710	0.9870	Correct
$\mathbb{E}[x_1^3]$	0.9902	0.001057	0.0000	Wrong (Stuck!)
$\mathbb{E}[x_1^4]$	0.9994	0.001412	0.9995	Correct

6 Conclusion

In this report, we investigated a range of Monte Carlo sampling methods for multimodal target distributions arising from nonconvex potential functions. Specifically, we studied the double-well and triple-well potentials

$$V(x) = (x^2 - 1)^2, \qquad V(x) = \frac{1}{40}(x^2 - 1)^2(x^2 - 4)^2(x^2 - 9)^2, \tag{21}$$

which serve as challenging yet representative benchmarks for sampling algorithms due to their well-separated modes and metastable regions.

We implemented and analyzed several classical and modern sampling techniques, including importance sampling, Metropolis–Hastings MCMC, annealing strategies (such as simulated tempering and parallel tempering), Gibbs sampling (with an extension to higher dimensions), and Langevin Monte Carlo. For each method, we provided detailed pseudocode and brief algorithmic analysis to highlight their underlying mechanisms and practical trade-offs.

All algorithms were implemented in Python using a unified and modular codebase, including standalone scripts, reusable class-based implementations, test functions, and example usage. The complete code was uploaded to the assigned Git repository to ensure reproducibility and ease of extension. Numerical results were presented through extensive visualizations, including histograms of sampled distributions, trace plots, and tables reporting moment estimates from first to fourth order. In addition, we conducted quantitative error analysis by plotting the root mean square error on a log–log scale as a function of the number of samples, and compared empirical convergence rates against the baseline $CN^{-1/2}$, where the normalization constant C ensures a common reference point.

The chosen potential landscapes are deliberately challenging: probability mass is highly concentrated near isolated wells, making transitions between modes rare. As a result, some methods—particularly those relying on acceptance–rejection mechanisms, such as annealing-based strategies—may suffer from stagnation when initialized poorly, since low acceptance probabilities can trap the chain within a single well for extended periods. In contrast, Gibbs sampling and Langevin Monte Carlo exhibit clear advantages in this setting. Rather than relying on discrete proposal selection, these methods update the state deterministically or gradient-driven along the energy landscape, which enforces movement and improves exploration of the state space.

Overall, this project provided hands-on experience with a diverse set of Monte Carlo methods and offered valuable insight into their behavior on multimodal targets. Through both theoretical analysis and practical implementation, we developed a deeper understanding of the strengths and limitations of different sampling

Table 7: Moment Estimates of Gibbs Sampling for $D = 20$ ($\beta = 20.0$)

Moment	**Estimate**	**Naive SEM (±)^1**	**True Value**	**Result**
$\mathbb{E}[x_1]$	0.9895	0.000366	0.0000	Wrong (Stuck!)
$\mathbb{E}[x_1^2]$	0.9858	0.000717	0.9870	Correct
$\mathbb{E}[x_1^3]$	0.9886	0.001066	0.0000	Wrong (Stuck!)
$\mathbb{E}[x_1^4]$	0.9976	0.001421	0.9995	Correct

strategies, reinforcing key concepts from Monte Carlo methods and scientific computing with Python.

References

[1] S. Ferson. What monte carlo methods cannot do. *Human and Ecological Risk Assessment: An International Journal*, 2(4):990–1007, 1996.

[2] W. Janke. Monte carlo methods in classical statistical physics. In *Computational many-particle physics*, pages 79–140. Springer, 2008.

[3] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev. Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):386–392, 2014.

[4] J. Machta. Strengths and weaknesses of parallel tempering. *Phys. Rev. E*, 80:056706, Nov 2009.

[5] P. Mocz. Create your own metropolis-hastings markov chain monte carlo algorithm for bayesian inference (with python), 2023.

[6] D. Sanz-Alonso and O. Al-Ghattas. A first course in monte carlo methods, 2024.

[7] S. T. Tokdar and R. E. Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.

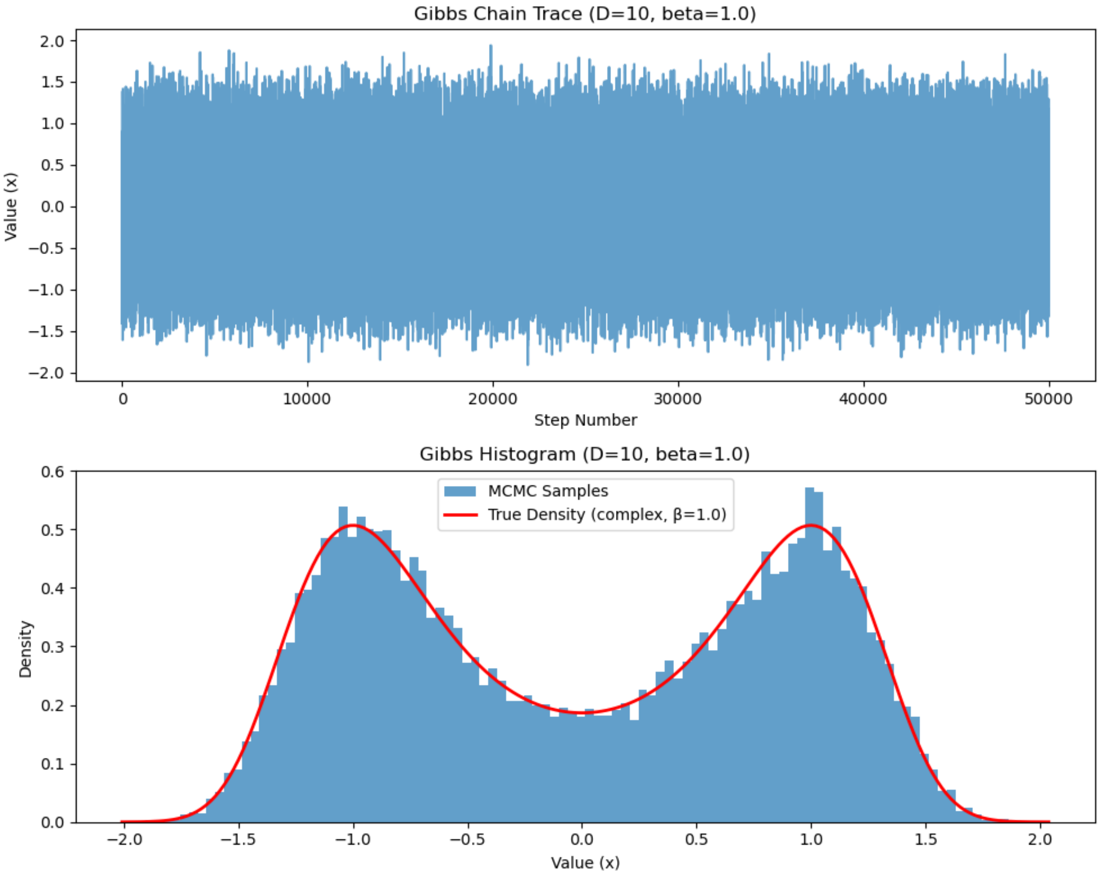

Figure 9: Gibbs Sampler (D=10, $\beta = 1.0$) trace and histogram. The chain successfully mixes across the modes.

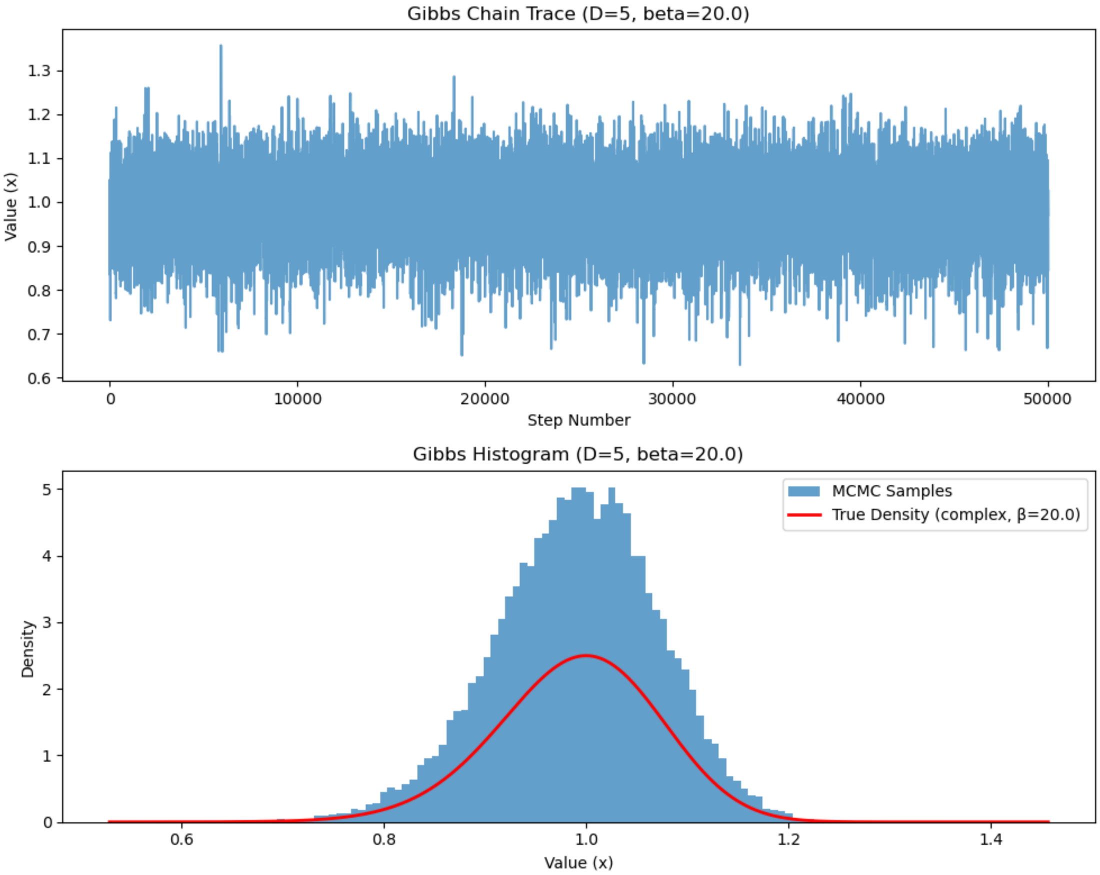

Figure 10: Gibbs Sampler (D=5, β = 20.0) trace and histogram. The chain is stuck in the positive mode, failing to sample the negative mode.

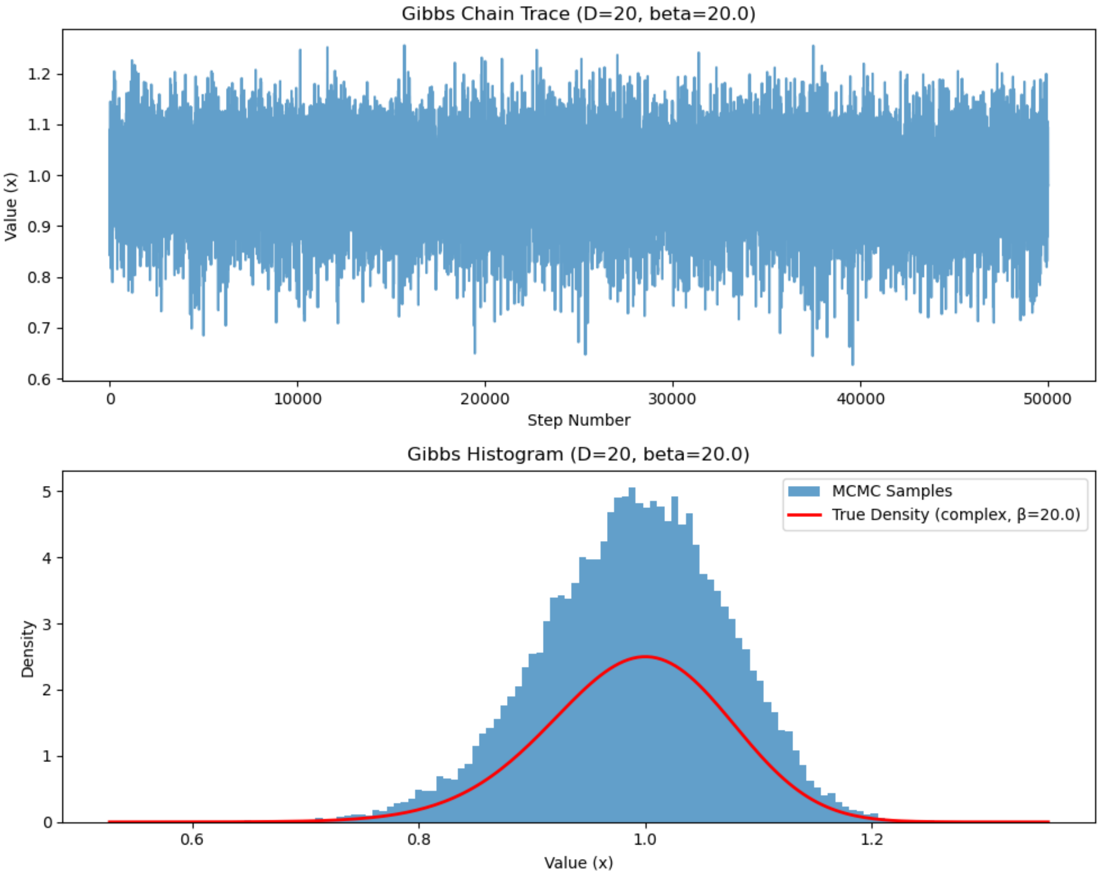

Figure 11: Gibbs Sampler (D=20, β = 20.0) trace and histogram. Increasing D does not improve mixing; the chain remains stuck, replicating the 1D M-H failure.

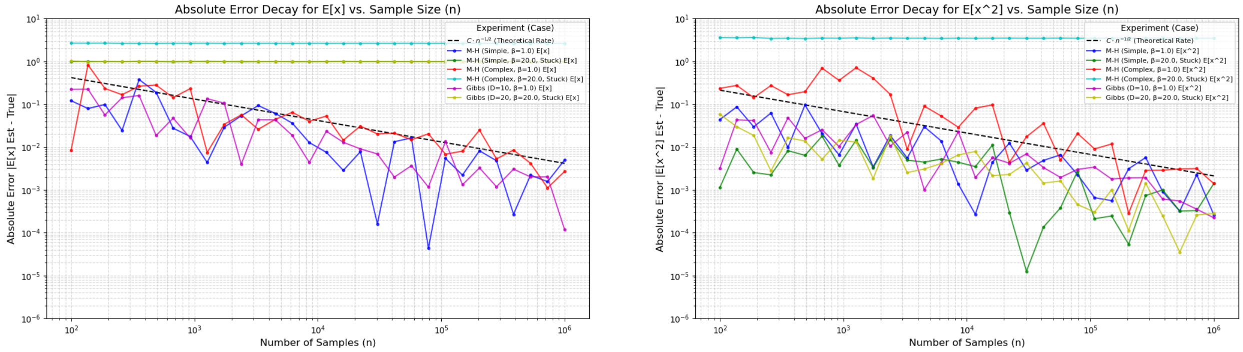

Figure 12: Convergence of the first two moments.

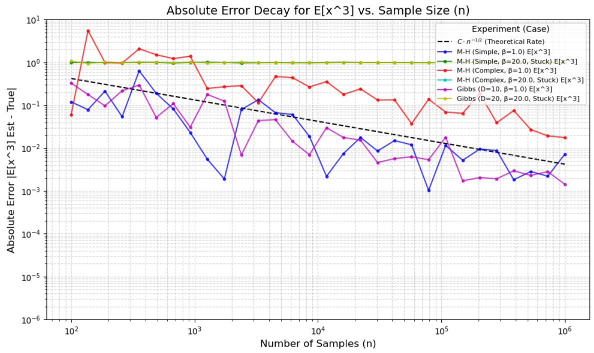

Figure 13: Convergence of the third and fourth moments.

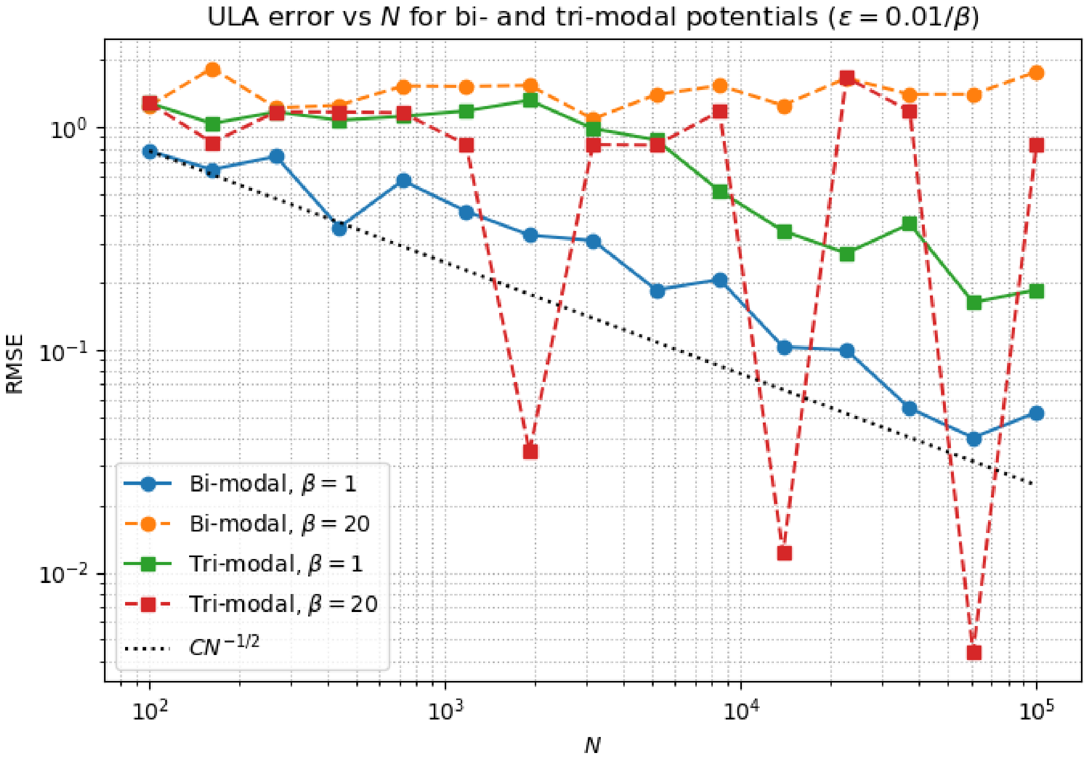

Figure 14: Root mean square error (RMSE) of Unadjusted Langevin Algorithm (ULA) estimators as a function of the sample size N for bi-modal and tri-modal potentials. Results are shown for inverse temperatures $\beta = 1$ and $\beta = 20$, with the stepsize scaled as $\varepsilon = 0.01/\beta$. The dashed reference line indicates the benchmark rate $\mathcal{O}(N^{-1/2})$.