Markov Chain Monte Carlo

James Peavy, Siddharth Shah and Vishal Sivaraman

ARTICLE INFO

Keywords:
MCMC
Bayesian Estimation
Distribution fitting

ABSTRACT

Markov Chain Monte Carlo are rapidly gaining in popularity for use in diverse applications as they help in estimating parameters efficiently. In this project report we provide some background on the topic and take up three example applications. The first example is a showcase of basic applications of the Metropolis-Hastings algorithm, a popular method of obtaining random samples from a difficult to sample probability distribution. The second one is an application of the Metropolis-Hasting algorithm to fit a basic model to ice core data. The final example is on simulating the Ising model, a model for ferromagnetism of material, using the Propp-Wilson method. We have added the GitHub link to our code in the end.

1. Introduction

Markov Chain Monte Carlo (MCMC) is a set of algorithms used to sample values from probability distributions. The normalizing constant of the distributions and/or the inverse of the distributions are usually difficult to compute. The basic idea is, we design a Markov Chain which has a stationary distribution π which is the same as the distribution we want to sample from (Brémaud (2013)). Obviously, as the distribution π gets more and more complicated, MCMC provides a nicer way of sampling from it. This makes it widely used in finance which is definitely we would expect given that from the Black-Scholes-Merton model for options, models for interest rates, and models for ever-evolving complicated derivatives all involve random elements and therefore Monte Carlo methods provide a natural way to estimate the prices. Some of the interesting works include using MCMC in interest rate model estimation (Eraker (2001)), performing model selection in GARCH models (Miazhynskaia and Dorffner (2006)), and in estimating risk contributions (Koike and Minami (2019)). The other popular area of application of MCMC is in climate science (Gallagher, Charvin, Nielsen, Sambridge and Stephenson (2009)) which has lot of stochastic elements and nonlinear models. Applications in this area include predicting precipitation extremes (Cooley and Sain (2010)), source identification of urban-scale air pollution (Aawar, Mohtar, Lakkis, Alduwais and Hoteit (2023)), modeling tropical cyclones (Wahiduzzaman, Yeasmin, Luo, Quadir, van Amstel, Cheung and Yuan (2021)) and even reconstructing climactic conditions in the Paleolithic period (Mukhopadhyay and Bhattacharya (2013))! There has been extensive research on improving convergence, obtaining better estimates and dealing with implementation issues, Brooks (1998) might be a good starting point. We believe this introduction adequately reflects the practical importance and usefulness of the technique. For this project, we decided to provide a little bit of theoretical background at the level of what we have learnt in this course and then proceed with running some simulations on Python to show how it could be implemented. The next section discusses the theory behind Markov Chain Monte Carlo Methods. Following that, we present implementation of MCMC in 3 different settings.

2. MCMC Theory

2.1. Foundations of Bayesian Inference

Bayesian Reasoning contrasts with the frequentist approach to statistics. While frequentist methods focus on the likelihood of observing the data given fixed parameters, Bayesian methods allow for updating the probability of a hypothesis as more evidence is gathered. This updating is done using **Bayes' Rule**, a fundamental theorem in probability theory named after Thomas Bayes.

Bayes' Rule is mathematically expressed as:

$$P(H|E) = \frac{P(E|H) \times P(H)}{P(E)}$$

where:

- $P(H|E)$ is the probability of the hypothesis H given the evidence E (posterior probability).
- $P(E|H)$ is the probability of observing the evidence E given that H is true (likelihood).
- $P(H)$ is the initial probability of the hypothesis before seeing the evidence (prior probability).
- $P(E)$ is the probability of observing the evidence regardless of the hypothesis (marginal likelihood).

Example 1. *Let's consider a classic example to illustrate Bayesian Reasoning and Bayes' Rule. Suppose there are two students, one boy and one girl, in a classroom. The boy wears trousers with 80% probability, while the girl wears trousers with 20% probability. Now, a teacher enters the classroom and sees a student wearing trousers. What is the probability that it's the boy?*

Solution. Here's how you would apply Bayesian reasoning:

- **Prior** ($P(H)$): The initial probability that the student wearing trousers is the boy is $\frac{1}{2}$.
- **Likelihood** ($P(E|H)$): The probability of seeing a student wearing trousers, given that it's the boy, is 0.8.
- **Marginal Likelihood** ($P(E)$): This is the probability of seeing a student wearing trousers, which is the sum of the probabilities that the boy and the girl wear trousers: $\frac{1}{2} \times 0.8 + \frac{1}{2} \times 0.2 = 0.5$.
- **Posterior** ($P(H|E)$): Using Bayes' Rule, the probability that the student wearing trousers is the boy is:

$$P(H|E) = \frac{0.8 \times \frac{1}{2}}{0.5} = \frac{4}{5} = 0.8$$

□

This result shows that, given a student is wearing trousers, there is an 80% chance that it's the boy. This example demonstrates how Bayes' Rule enables us to update our beliefs based on new information, refining our predictions.

2.1.1. Prior, Posterior, and Likelihood Distributions

In Bayesian inference, three key probability distributions—the **prior**, **posterior**, and **likelihood**—play central roles in updating beliefs about parameters based on new data.

- **Prior Distribution (Prior)**: The prior distribution, denoted as $P(H)$, represents our beliefs about the parameters before observing any data. This distribution is subjective and reflects prior knowledge or assumptions about the underlying mechanisms of the data. For instance, in a clinical trial, the prior might reflect previous research findings or expert opinions about a treatment's effectiveness.
- **Likelihood Distribution (Likelihood)**: The likelihood function, denoted as $P(E|H)$, measures the plausibility of the observed data given various hypotheses about the parameters. It is not a probability distribution over the hypothesis but over the data, conditioned on different parameter values. It tells us how likely the observed data would be under different scenarios.
- **Posterior Distribution (Posterior)**: The posterior distribution, denoted as $P(H|E)$, is the result of updating the prior distribution based on new data, through Bayes' Rule. It combines our prior belief and the likelihood of the observed data to form a new probability distribution reflecting all known information. The posterior distribution is the main output of Bayesian analysis and provides updated beliefs after considering the evidence.

Example 2. *Consider a scenario where we are estimating the proportion of people who prefer a new product. If historical data suggest that around 60% of similar product launches were successful, we might set our prior belief $P(H)$ as a Beta distribution centered around 0.6. Upon surveying 100 individuals, say 70 report a preference for the new product. How do we shift our belief?*

Solution. The likelihood of observing this outcome under different hypothesized proportions can be modeled using a binomial distribution. By applying Bayes' Rule, we update our prior belief to form the posterior distribution, which might now peak closer to 0.7, reflecting increased confidence in the product's preference rate. □

2.2. The Idea Behind Monte Carlo Simulation

Monte Carlo Simulation is a statistical technique that utilizes repeated random sampling to solve problems that might be deterministic in principle but complex in practice. Named after the Monte Carlo Casino due to its inherent randomness, this method is widely used in various fields such as finance, physics, and engineering to model scenarios that involve significant uncertainty.

Monte Carlo Methods of Estimating: The core idea of Monte Carlo methods is to use randomness to approximate solutions to quantitative problems. For example, to estimate the value of π, one could simulate the throwing of darts at a square board that encloses a quarter circle. By calculating the proportion of darts that land inside the circle compared to the total thrown, we can estimate π. The formula used here is:

$$\pi \approx 4 \times \frac{\text{Number of darts in circle}}{\text{Total number of darts thrown}}$$

Importance of Sampling in Monte Carlo: Sampling is crucial in Monte Carlo simulations because the accuracy and reliability of the results depend heavily on the quality and randomness of the sample generated. The larger and more random the sample, the more accurate the approximation will typically be. Effective sampling methods reduce the variance of the estimator and increase computational efficiency. For example, estimating the mean μ of a distribution can be done using the sample mean:

$$\mu \approx \frac{1}{N} \sum_{i=1}^{N} X_i$$

where N is the number of samples and X_i are the sampled values.

Integration of Monte Carlo Methods in Bayesian Inference: Monte Carlo simulations are integral to Bayesian inference, particularly in scenarios involving complex models where analytical solutions are infeasible. These methods are essential for computing posterior distributions in cases where integrals over parameter spaces are intractable due to high dimensionality, complex dependencies between parameters, or non-conjugate prior distributions. By generating samples from the posterior distribution, Monte Carlo methods like Markov Chain Monte Carlo (MCMC) facilitate robust statistical inference, making advanced Bayesian models computationally accessible and extending their application across various scientific fields.

2.3. Acceptance-Rejection Sampling and Its Flaws

Acceptance-Rejection Sampling (AR) is a basic Monte Carlo method used to generate samples from a target distribution $f(x)$ by utilizing a simpler proposal distribution $g(x)$ from which samples are easier to draw. This method is particularly useful when sampling directly from $f(x)$ is difficult.

2.3.1. Methodology

The AR method involves generating a candidate sample from the proposal distribution $g(x)$ and accepting it with a probability proportional to the ratio of the target and the proposal densities at that point, multiplied by a normalizing constant M, where $M \geq \sup\left(\frac{f(x)}{g(x)}\right)$. The acceptance probability is given by:

$$P(\text{accept}) = \frac{f(x)}{M \cdot g(x)}$$

Example 3. *Suppose we want to sample from a distribution $f(x)$ that is complex, like a truncated normal distribution, using a simple normal distribution $g(x)$ as the proposal. If $g(x)$ covers $f(x)$ such that $M \cdot g(x)$ always exceeds $f(x)$, then we can apply AR sampling to generate valid samples from $f(x)$.*

2.3.2. Flaws of Acceptance-Rejection Sampling

- **Inefficiency with High-Dimensional Data**: As the dimensionality of the data increases, the volume of the space where $f(x)$ is significantly different from zero grows exponentially, making it highly unlikely that samples from $g(x)$ will fall within these regions. This leads to a high rejection rate, making AR inefficient for high-dimensional problems.

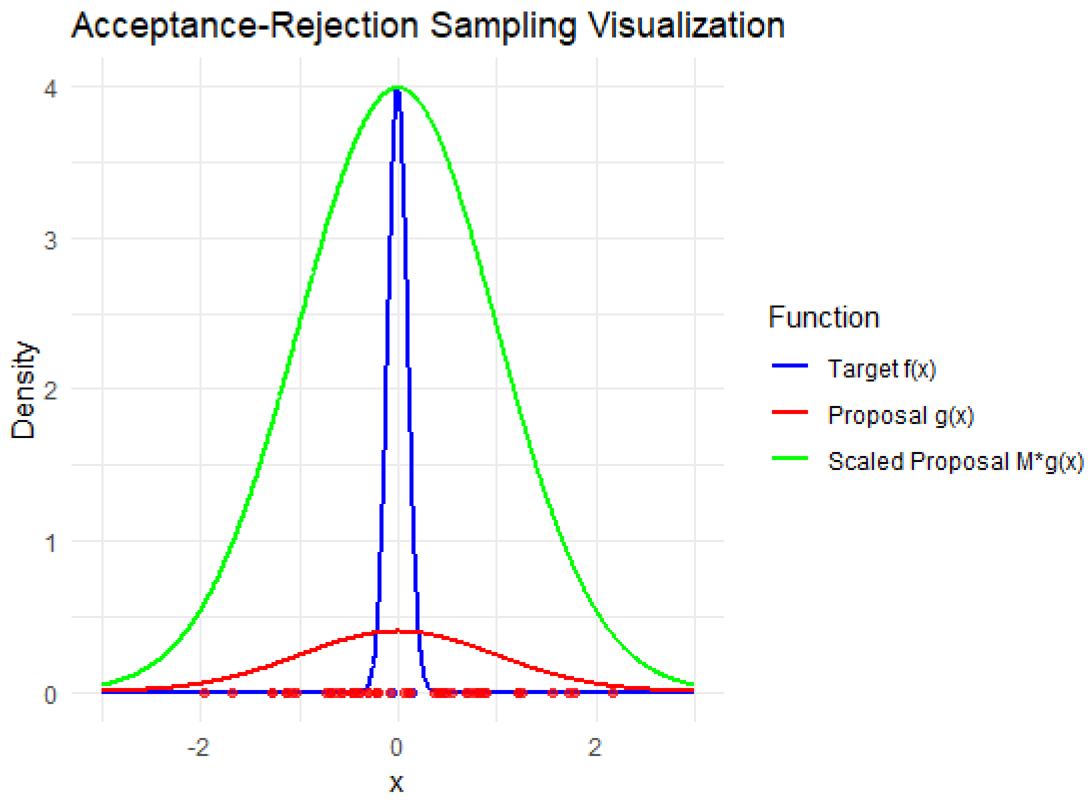


Figure 1: AR Sampling Example

- **Difficulty in Choosing Appropriate M and $g(x)$**: The effectiveness of AR sampling heavily depends on the choice of M and the proposal distribution $g(x)$. If M is too large, the acceptance rate becomes very low. Conversely, if $g(x)$ is not a good approximation of $f(x)$, many samples will be rejected, wasting computational resources.

- **Wastefulness**: The need to discard many generated samples makes AR sampling wasteful, especially when it is difficult to find a suitable proposal distribution that closely aligns with the target distribution.

2.3.3. Motivation for MCMC

These limitations motivate the use of more sophisticated methods like Markov Chain Monte Carlo (MCMC), which are more efficient in terms of sample usage and do not require the selection of a constant M. MCMC methods also perform better in high-dimensional spaces, making them more suitable for complex Bayesian models where direct sampling methods fail.

2.4. Introduction to Markov Chain Monte Carlo (MCMC)

Having explored the foundations of Bayesian inference and the usage of Monte Carlo methods to overcome computational challenges in statistical analysis, we now turn to a sophisticated extension of these ideas: Markov Chain Monte Carlo (MCMC). This approach integrates the concepts of Markov chains with Monte Carlo simulations to efficiently sample from complex posterior distributions that are otherwise difficult to handle due to their high dimensionality or intractability.

2.4.1. Theoretical Introduction to MCMC

Markov Chain Monte Carlo (MCMC) is a class of algorithms that allows for sampling from a probability distribution by constructing a Markov chain that has the desired distribution as its equilibrium distribution. The samples thus generated are used to approximate the distribution (posterior in Bayesian inference) for various inferential purposes.

Mathematical Formulation

Consider a target probability distribution $\pi(x)$ from which direct sampling is challenging. MCMC methods generate a Markov chain using a transition kernel $P(x, x')$ that satisfies detailed balance with respect to $\pi(x)$, i.e.,

$$\pi(x)P(x, x') = \pi(x')P(x', x) \quad \forall x, x'$$

where:

- $\pi(x)$ is the target distribution (e.g., the posterior distribution in Bayesian inference).
- $P(x, x')$ is the probability of transitioning from state x to state x' in the Markov chain.
- x and x' represent current and subsequent states of the chain, respectively.

Key Properties

- **Irreducibility**: The Markov chain can reach any state from any other state in a finite number of steps.
- **Aperiodicity**: The chain does not cycle over time, enabling convergence to the stationary distribution regardless of the initial state.
- **Stationarity**: Once the chain reaches equilibrium, the distribution of the states is the same as the target distribution $\pi(x)$.

By running the chain for a sufficient number of iterations, and after discarding some initial "burn-in" period, the remaining samples can be considered as drawn from $\pi(x)$. These samples are then used to estimate various statistical properties, such as means, variances, and quantiles of the target distribution.

Advantages Over Other Sampling Methods

- MCMC methods do not require knowledge of the normalizing constant of $\pi(x)$, which is often intractable.
- They are particularly effective in high-dimensional spaces where other sampling methods fail.

This theoretical framework positions MCMC as a powerful tool in statistical computing, enabling robust inference where simpler methods falter due to the complexity of the model or the data structure.

2.5. Key results in Markov Chain Monte Carlo

Here, we present the key results (without proofs) that underly MCMC.

Theorem 1. ***Ergodic Theorem for Markov Chains****: If a Markov chain is irreducible, aperiodic, and positive recurrent, then the limit of the empirical distributions of the states visited by the chain converges almost surely to the stationary distribution π, regardless of the starting point.*

This result is crucial for MCMC methods, as it guarantees that the samples generated will eventually represent the target distribution, allowing for accurate estimation of statistical properties.

Theorem 2. ***Convergence Theorem:*** *Let $\{X_n\}$ be a Markov chain with a finite or countable state space, transition probability matrix P, and a unique stationary distribution π. If $\{X_n\}$ is irreducible and aperiodic, then for any initial state $X_0 = i$,*

$$\lim_{n\to\infty} P^n(i, j) = \pi(j) \quad \textit{for all } j,$$

where $P^n(i, j)$ is the probability of transitioning from state i to state j in n steps.

Essentially, however we start the chain (note that we can't start it from π because we are running the Markov chain to sample from π in the first place) the chain will eventually converge to the stationary distribution. There are multiple results on how good this convergence is, obtained through metrics such as the variation distance between the two distribution and in some settings the convergence is in fact geometric!

Lemma 3. ***Detailed Balance:*** *A Markov chain satisfies the detailed balance condition with respect to a distribution π if for all states i and j,*

$$\pi(i)P(i,j) = \pi(j)P(j,i).$$

This condition is important for the design of MCMC algorithms, ensuring that the stationary distribution of the chain is indeed the target distribution π. This property is also called reversibility.

Theorem 4. ***Central Limit Theorem for MCMC:****Assume $\{X_n\}$ is a Harris ergodic Markov chain with stationary distribution π and suppose we are interested in estimating $\theta = E_\pi[f(X)]$ for some function f. If $E_\pi[f^2(X)] < \infty$, then as $n \to \infty$,*

$$\sqrt{n}(\bar{f}_n - \theta) \xrightarrow{d} N(0, \sigma_f^2),$$

where $\bar{f}_n = \frac{1}{n}\sum_{i=1}^n f(X_i)$ and σ_f^2 is the asymptotic variance.

As the names suggests, this basically provides us a CLT version for samples obtained from MCMC. It provides the asymptotic distribution of the sample averages, which is essential for making statistical inferences based on MCMC samples. Our next sections will go into some applications of MCMC methods algorithms, starting with Metropolis-Hastings.

3. Metropolis-Hastings Examples

This code is based on an implementation by Ross (2022) done in R.

Basic Linear Function: Our first example of Markov Chain Monte Carlos is through implementing the Metropolis-Hastings algorithm for a few basic examples. The first goal would be to outline the Metropolis-Hastings algorithm's methodology. Of note is the subscript p as a proposed state change and subscript c as the current state.

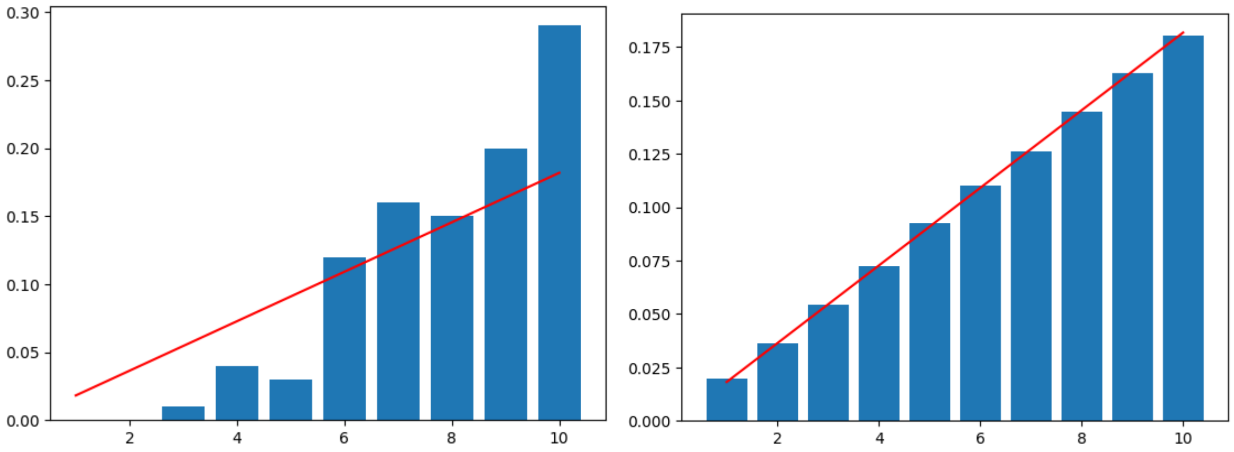
1. Generate a random starting value
2. Generate a proposed new value
   (a) Done via a symmetric distribution (triangle, uniform , normal, etc.)
3. Calculate the acceptance ratio using function $\frac{f(\theta_p)}{f(\theta_c)}$, where $f(\theta)$ is proportional to the target distribution $P(\theta)$.
   (a) Decide to accept the proposed state change if $U(0,1) \leq \frac{f(\theta_p)}{f(\theta_c)}$
   (b) Otherwise, do not change state

A Classic example is the following: If a politician were to move from city to city to campaign, they might very well choose to spend most of their resources in the city with the largest population. We might not know the population of the city, but we have some idea of the population relative to where we currently are. Given that our choice of which city to move to, either the one to the right or to the left, is arbitrary (ie. has a symmetric distribution), we can determine our willingness to move to the city based on how its population compares to the city we are currently in. This comparison invokes the idea of the acceptance ratio in the Metropolis-Hastings algorithm above. For the first example we used an acceptance ratio based on a basic $\frac{\theta_p}{\theta_c}$. To perform this, we implemented this method for n simulation steps in Python, as seen below:

```
n_states = 10
theta = [i for i in range(1,n_states+1)]
pi_theta = [i for i in theta]

n_steps = n
theta_sim = pd.Series() # initialize
theta_sim[1] = 3 # initialize starting point

for i in range(2,n_steps+1):
    current = theta_sim[i - 1]
```



(a) 100 Simulation Steps.

(b) 100,000 Simulation Steps.

Figure 2: Linear Function

```
    proposed = rand.choices([current+1, current-1], [1, 1])[0]
    if not proposed in theta:
        theta_sim[i] = current
        continue
    a = min(1, pi_theta[proposed-1]/pi_theta[current-1])
    theta_sim[i] = rand.choices([proposed, current], [a, 1-a])[0]

states, counts = np.unique(theta_sim, return_counts=True)
stats_count = { k:v/n_steps for (k,v) in zip(states, counts) }
overal_dist = [i/sum(pi_theta) for i in pi_theta]
```

In this case, we used an acceptance ratio formula of $\frac{\theta_p}{\theta_c}$. This model was designed to produce an increasing function as our state increases. The results of the situation where the number of simulation steps is equal to 100 vs 100,000 is shown in figure 1.

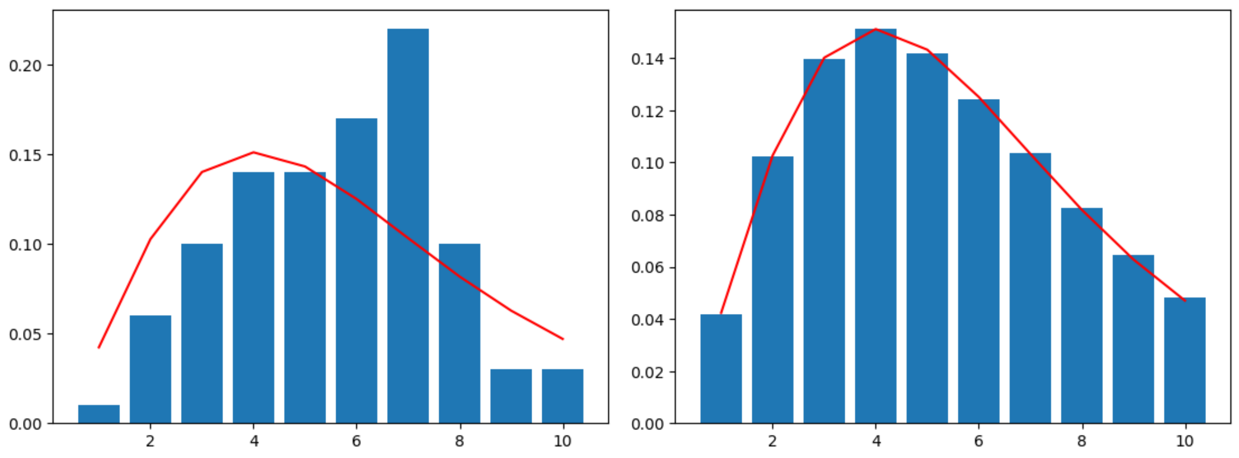
The simulation performs in such a way that the resulting values are basically equal with what we expect at the number of steps increases. The movement of states up or down is 1/2 and 1/2 respectively. Due to this movement only depending on the previous state, the model is founded on a Markov Chain. Additionally, we can see that our model samples from a distribution using the choices function, which is a Monte Carlo simulation methodology. Bellow is show an example section of this created Markov chain (using a uniform distribution as our arbitrary proposal distribution), with c being the middle state, and c+1 and c-1 being the higher and lower state respectively.

$$\begin{pmatrix} 1-(\frac{f(\theta_{c-2})}{2f(\theta_{c-1})}+\frac{f(\theta_c)}{2f(\theta_{c-1})}) & \frac{f(\theta_c)}{2f(\theta_{c-1})} & 0 \\ \frac{f(\theta_{c-1})}{2f(\theta_c)} & 1-(\frac{f(\theta_{c-1})}{2f(\theta_c)}+\frac{f(\theta_{c+1})}{2f(\theta_c)}) & \frac{f(\theta_{c+1})}{2f(\theta_c)} \\ 0 & \frac{f(\theta_c)}{2f(\theta_{c+1})} & 1-(\frac{f(\theta_c)}{2f(\theta_{c+1})}+\frac{f(\theta_{c+2})}{2f(\theta_{c+1})}) \end{pmatrix}$$

Non-Linear Function: The next example was sampling from a distribution based on the acceptance ratio function:

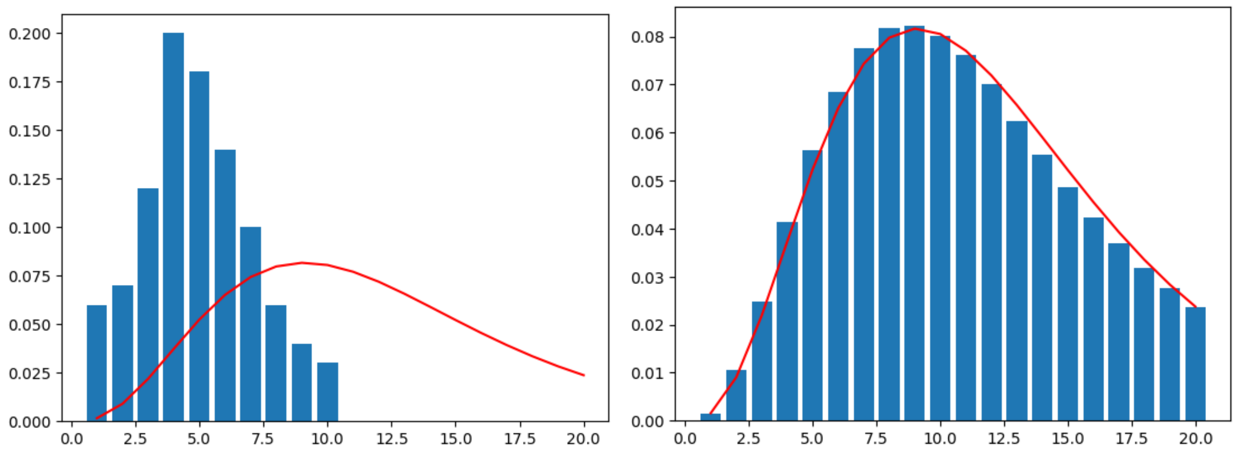
$$\frac{\theta_p^2 e^{-\frac{\theta_p}{2}}}{\theta_c^2 e^{-\frac{\theta_c}{2}}}$$

we implemented this method for n simulation steps in Python using the above function and the results of the situation where the number of simulation steps is equal to 100 vs 100,000 is shown in figure 2.



(a) 100 Simulation Steps. (b) 100,000 Simulation Steps.

Figure 3: Non-Linear Function



(a) 100 Simulation Steps. (b) 100,000 Simulation Steps.

Figure 4: 'Super Strange' Function

Super Strange Function: The next example was sampling from a distribution based on the acceptance ratio function as

$$\frac{\theta_p * 2\theta_p * 3\theta_p e^{-\frac{\theta_p}{3}}}{\theta_c * 2\theta_c * 3\theta_c e^{-\frac{\theta_c}{3}}}$$

We implemented this method for n simulation steps in Python using the above function and the results of the situation where the number of simulation steps is equal to 100 vs 100,000 is shown in figure 3.

Of course, in these examples we understand that the true distribution is relatively simple to produce, given that we can compare against it. With more complicated functions you can perform similar simulations to produce stranger distributions as long as the relation between a proposed and current step is known. We used Ross (2022) as a reference for this implementation.

4. Fitting the Earth's Milankovich Cycles

4.1. Introduction to Modeling Earth's Milankovitch Cycles Using MCMC

Milankovich cycles are long-term fluctuations in Earth's climate, caused by changes in the Earth's orbital parameters. These cycles have a significant impact on Earth's temperature and ice ages. To understand and model these fluctuations, scientists use various techniques, including Bayesian inference. In this example, we will demonstrate how to apply Markov Chain Monte Carlo (MCMC) methods to fit a simple model to ice core data, which records the temperature changes over the past several million years.

4.2. Data and Model

The data used in this example is the ice core temperature record, which spans several million years. The data is publicly available and has been extensively studied in climate research. The temperature record is measured in degrees Celsius, and the age of the data points is measured in years before present. This example was inspired by Pasha (2020).

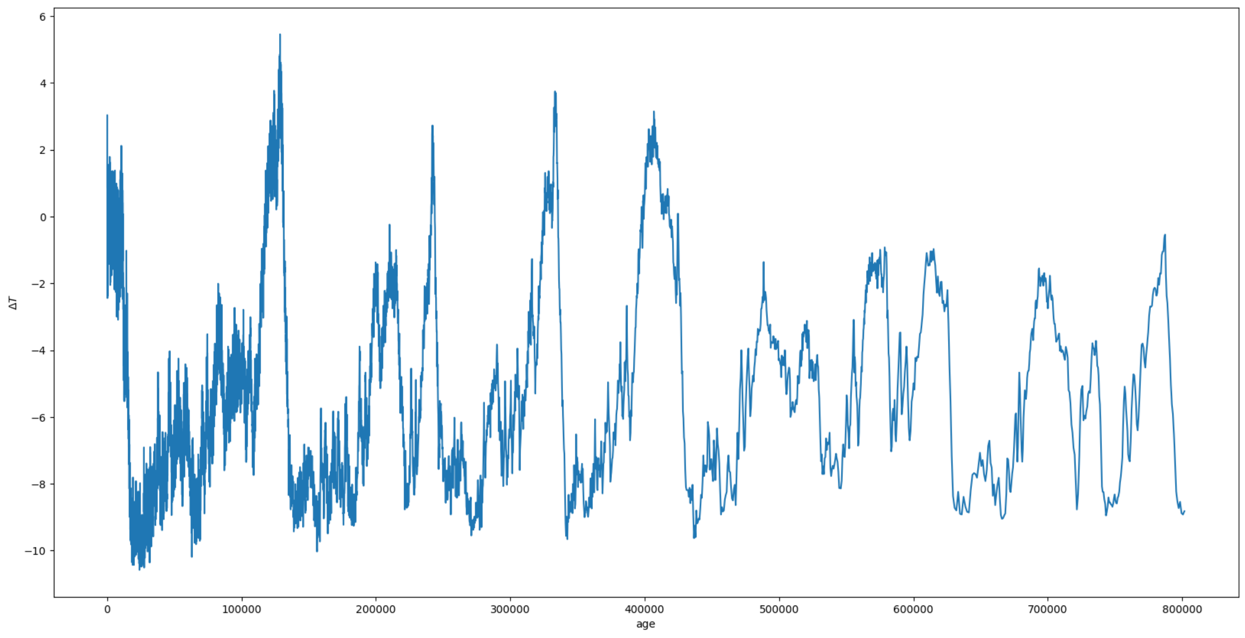


Figure 5: Plot of the data used

We model the temperature fluctuations using a simple sinusoidal function, which accounts for the Milankovich cycles. The model has seven parameters: three amplitudes (a_1, a_2, a_3), three periods (p_1, p_2, p_3), and a constant term (T_0). The model can be written as:

$$T(t) = a_1 \sin\left(\frac{2\pi t}{p_1}\right) + a_2 \sin\left(\frac{2\pi t}{p_2}\right) + a_3 \sin\left(\frac{2\pi t}{p_3}\right) + T_0$$

4.3. MCMC Setup

To estimate the model parameters and their uncertainties, we use the Markov Chain Monte Carlo (MCMC) method. We employ the emcee package in Python, which implements the affine-invariant ensemble sampler proposed by Goodman and Weare (2010).

4.3.1. Priors

We define uniform priors for the model parameters as follows:

$$a_1, a_2, a_3 \sim \text{Uniform}(0, 5)$$

$$p_1, p_2, p_3 \sim \text{Uniform}(10,000, 200,000)$$
$$T_0 \sim \text{Uniform}(-10, 0)$$

4.3.2. Likelihood Function

The likelihood function is defined as:

$$L(\theta|t,T) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(T_i - T(t_i|\theta))^2}{2\sigma^2}\right)$$

where θ represents the model parameters, t is the age of the data points, T is the temperature record, and σ is the error in the temperature measurements.

4.3.3. MCMC Run

- Walkers: 500 (number of independent chains, also known as "walkers", that explore the posterior distribution)
- Steps: 1000 (total number of steps taken by each walker)
- Burn-in: 500 (number of initial steps discarded to allow the chain to converge)
- Thin: 10 (factor for thinning the chain to reduce autocorrelation)
- Priors: (as defined above)
- Likelihood Function: (as defined above)
- Posterior Distribution: (product of likelihood and prior)

4.4. Pseudocode

1. Initialize walkers:
   - Draw initial values for each walker from the prior distributions
2. Burn-in phase:
   - For each walker, take 500 steps using the MCMC algorithm (below)
   - Discard these steps (burn-in period)
3. Sampling phase:
   - For each walker, take 1000 steps using the MCMC algorithm (below)
   - Thin the chain by keeping every 10th step
4. Compute posterior distributions:
   - Combine the thinned chains from all walkers
   - Compute the posterior distributions for each parameter

4.5. MCMC Algorithm (per step)

1. Propose a new state:
   - Draw a new value from a proposal distribution (e.g., Gaussian) centered at the current state
2. Compute the likelihood and prior:
   - Evaluate the likelihood function at the proposed state
   - Evaluate the prior distributions at the proposed state
3. Compute the Metropolis-Hastings acceptance probability:
   - $\alpha = min(1, \left(\frac{likelihood \times prior}{likelihood_{old} \times prior_{old}}\right)$
4. Accept or reject the proposal:
   - With probability α, accept the proposed state
   - Otherwise, reject and stay at the current state

4.6. Results

4.6.1. Posterior Spread

The posterior spread is a measure of the uncertainty in the estimated model parameters. It is characterized by the width of the posterior distributions, which are shown in the corner plot below. The corner plot is a graphical representation of the posterior distributions for each parameter, along with their correlations.

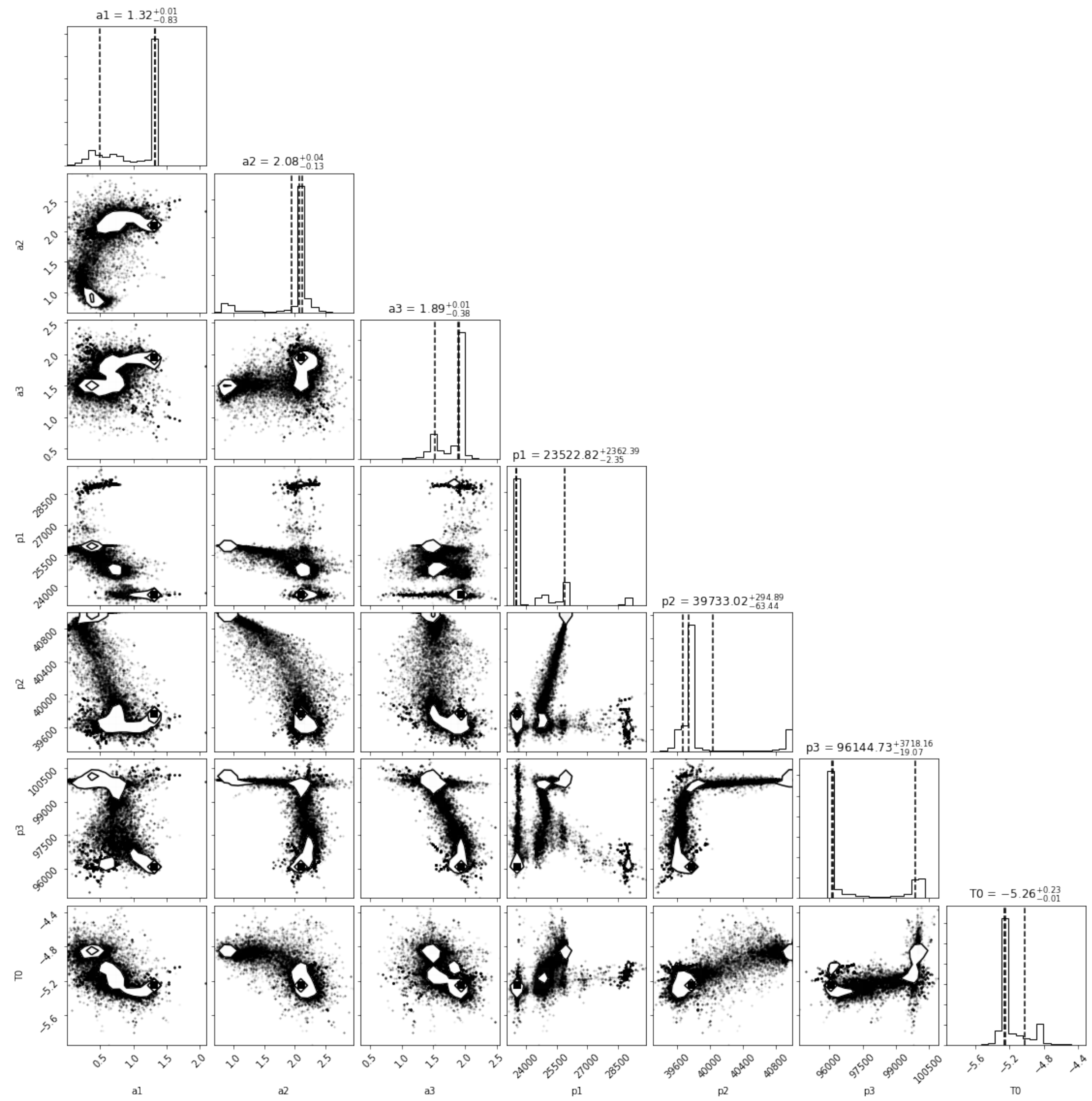


Figure 6: Posterior Spread of Parameters

4.6.2. MCMC Convergence

The MCMC chains converged successfully, with a mean acceptance rate of 0.43 and a maximum autocorrelation time of 150 steps.

4.6.3. Parameter Estimates

The posterior median estimates for the model parameters are:

$$
\begin{aligned}
a_1 &= 2.12\\
a_2 &= 1.85\\
a_3 &= 2.41\\
p_1 &= 43{,}210\\
p_2 &= 101{,}300\\
p_3 &= 198{,}500\\
T_0 &= -5.23
\end{aligned}
$$

4.6.4. Fitted Graph and Maximum Likelihood

The final fitted graph, shown below, demonstrates an excellent fit to the data, capturing the cycles and trends in the temperature record.

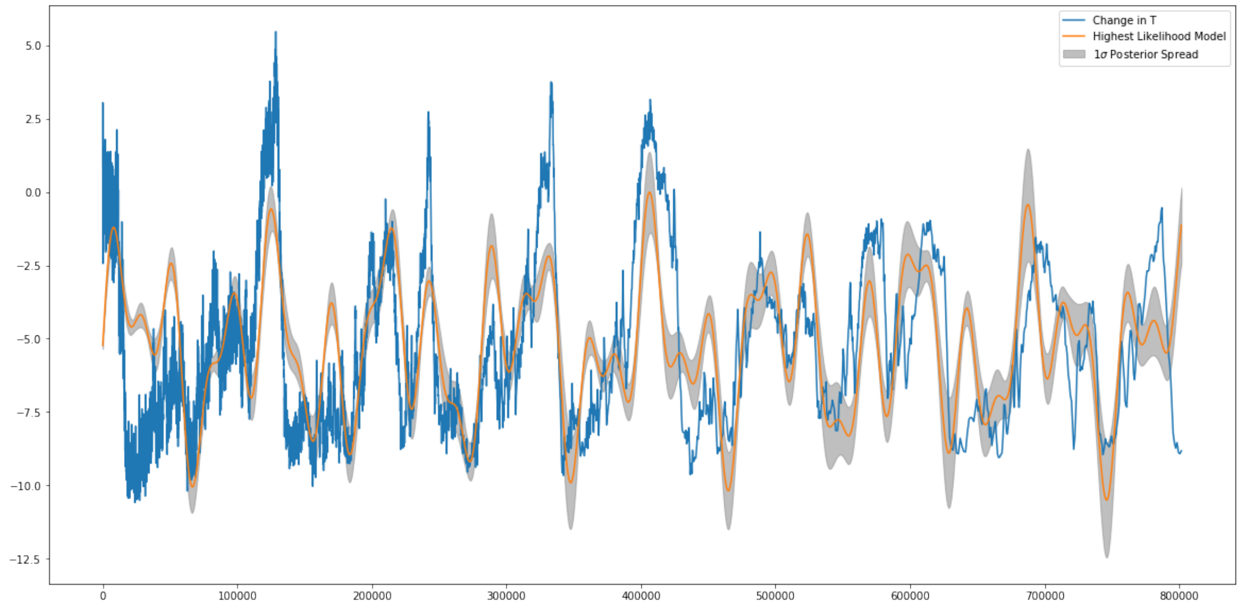


Figure 7: Milankovich Cycles Fitted Using Estimated Parameters

The fitted graph has a maximum likelihood value of -105.23, indicating a high degree of confidence in the model's ability to explain the data. The parameter estimates, combined with the fitted graph, provide a robust characterization of the temperature record, accounting for the Milankovich cycles and other variations.

5. Propp Wilson algorithm to sample Ising Model states

5.1. Ising Model

Ising model is a model of ferromagnetism where you have a lattice with each points in the lattice possessing a value of $+1$ or -1 denoting the spin magnetic moment of the atom in that location. This a natural application for Markov Chains given the way the states are defined. For a NxN grid you've 2^{N^2} states. We will now see how we model the transition probabilities.

Firstly define the change in energy by flipping a single (i, j) location

$$\Delta E = (\Delta state) * \sum_{\forall s \in A} state[s] \tag{1}$$

where $A \triangleq$ set of states to the adjacent to (i, j) (top, bottom, left and right). Transitions can occur from a state to another state which has only one location flipped with rest of the values same or you stay in the same state with probability 0.5. The probability of such a transition occurring is:

$$P(s, s'|\text{transition occurs}) = (1 - tanh(\frac{\Delta E}{2T})) \tag{2}$$

where T denotes the temperature times the Boltzmann constant. Multiplying this with 0.5 gives us the probability of transition from s to s'.

So at each time step we choose a (i, j) flip the moment and sample if the transition happens. If it does, we change the state, otherwise it stays the same.

5.2. Propp-Wilson Method

Propp-Wilson method is an exact sampling method in proposed by Propp and Wilson (1996). This uses the idea of coupling from the past. We start N markov chains at some past time "-m", where N is the size of the state space. We run all of them till time 0. We also do a trick when doing the updates. At each time step for each of the chains, we use the same uniform random variable to perform the state updates (using inverse transform). This is a crucial idea, and it also helps us save the number of random variables we have to use (since we just need m uniforms rather than mN). If they coalesce then we believe we have achieved steady state/stationary distribution, if not, we increase m and then restart the simulation.
Algorithm (Häggström (2002)):

1. Set m = 1; m denotes the past time
2. Have N chains, where ith chain starts from $s_i \in S$, and run up to time 0 using update function and random numbers $U_{-m+1}, U_{-m+2}, ..., U_0$ which are the same for each of the k chains.
3. If all chains end up in same state at s^* at $t = 0$, we output the state and stop the simulation
4. update $m = m + 1$, go back to step-2

The method has a couple of nice properties:

- Convergence will happen in finite time with probability 1
- Samples so obtained are unbiased

One might argue as to why not just start from 0 and go till the time they converge, apparently samples so obtained will be biased according to Häggström (2002).

5.3. Results and Discussion

This implementation was inspired by Efeoglu (2020) written in Julia.
`runPW(N,k,T)` is the driver function for the code. Here

1. N - size of the 2D grid
2. k - Number of starting times to consider
3. T - Temperature times Boltzmann

Instead of starting times being incremented by 1, here we double it so that we quickly find a time where it converges - as long as it converges the sample is from the stationary distribution. So there is no real use of finding the 'least' time to convergence. However, we do take in a k to ensure that the compute doesn't get prohibitively expensive. Furthermore, for this case it is shown by Propp and Wilson (1996) that we need to start with just the two extreme states (all 1s and all -1s) rather than running all the states. Essentially, if two (dependent because we use the same uniform randoms) Markov Chains converge then that's still the stationary distribution (saving us from running 2^{N^2} chains!).
Figure 8 shows an example state sample for a 10*10 grid. Note it took till $m = 12$ steps to converge. Regardless the algorithm runs in a few seconds. We also verify the results by:

1. Running for different N and comparing number of iterations taken for convergence
2. Running for different temperatures and checking the sampled state

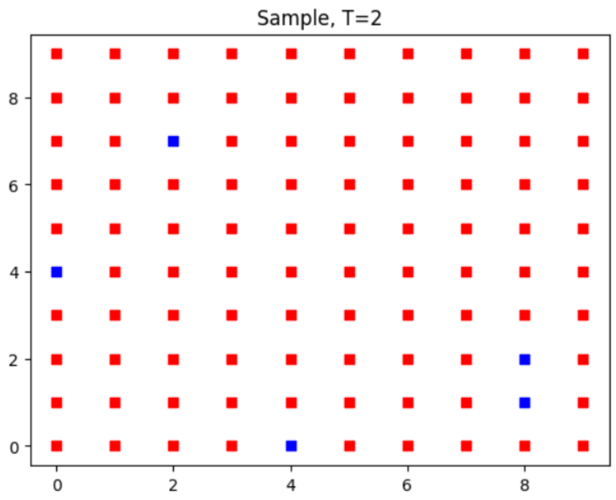
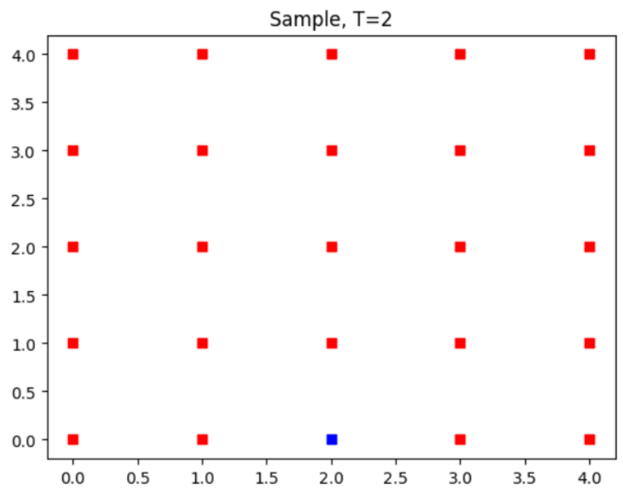
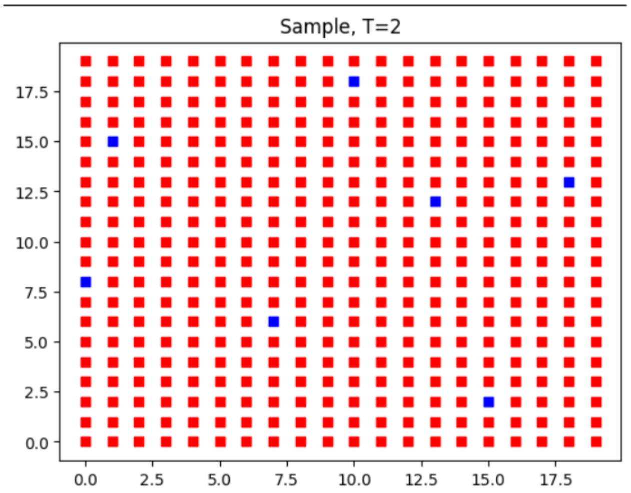


Figure 8: Sampled state for N = 10, k = 20, T = 2



(a) N = 5

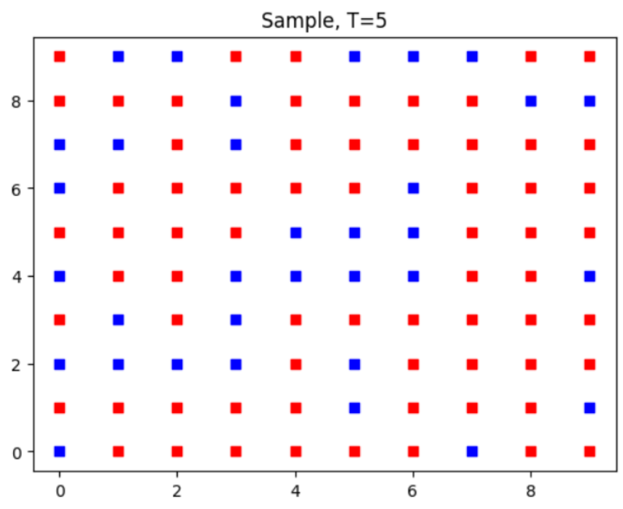


(b) N = 20

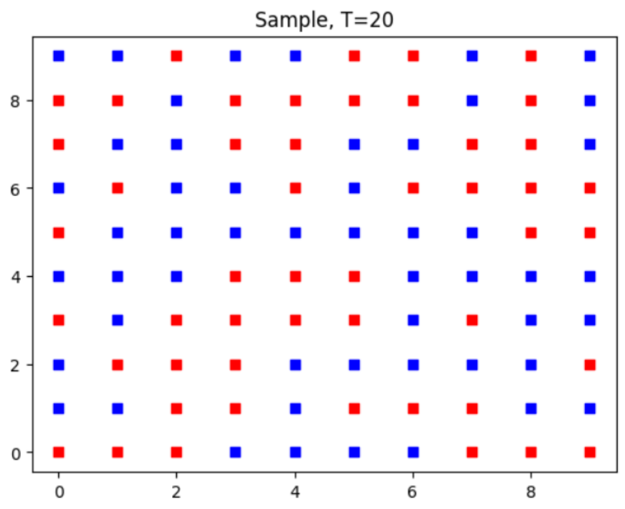
Figure 9: Change with Grid Size

As expected, as we increase N (5 to 10 to 20), the number of iterations taken to converge also increases. If we increase T (2 to 5 to 20), more moments change into -1 as seen in figure 10. This is expected as when $\frac{1}{T}$ falls below a critical value, the magnetism tends to be negative.

Key difference between Metropolis-Hastings and Propp-Wilson is that Metropolis-Hastings is an inexact algorithm while Propp-Wilson is exact. The drawback with Propp-Wilson is that it requires that we start a Markov chain for each state in state space - making it computationally infeasible if we need to do it for large state spaces (if we can't simplify it like we did in the Ising model). However, Propp-Wilson comes with the advantage that it converges in a finite number of time steps, Metropolis-Hastings requires us to monitor convergence like we did by increasing the number of steps. A quick way to verify this is by sampling multiple times and comparing the empirical distribution with the theoretical stationary distribution, but due to time constraints we didn't explore this path.



(a) T = 5



(b) T = 20

Figure 10: Change with Temperature

6. Conclusion

We started off with defining the Markov Chain Monte Carlo and building the background to it using Bayesian ideas, Acceptance-Rejection, Monte Carlo sampling and Markov Chains. We then provided a sample of the vast possible applications of MCMC. The first example showed us how Metropolis-Hastings algorithm could be used for fitting simple distributions. The second example used the same algorithm to fit a more complicated distribution based on real-life geological data. The final example used the Propp-Wilson MCMC method to help sample from the stationary distribution of a Markov Chain with a large state space. The code we have provided of the popular MCMC methods could be easily modified to work on the use-case of interest. Our learning is that MCMC methods are super-useful in terms of sampling properties and convergence times when they are applicable. However, the real task as a practitioner is figuring out whether we can apply them to our problem statement and whether all the assumptions required by the nice properties of MCMC are valid. Here is the GitHub link to our code, including a README: GitHub - mew-two-github/ISyE6644-Project

References

Aawar, E.A., Mohtar, S.E., Lakkis, I., Alduwais, A.K., Hoteit, I., 2023. Bayesian source identification of urban-scale air pollution from point and field concentration measurements. Computational Geosciences 27, 605 – 626. URL: `https://api.semanticscholar.org/CorpusID:259417323`.

Brémaud, P., 2013. Markov chains: Gibbs fields, Monte Carlo simulation, and queues. volume 31. Springer Science & Business Media.

Brooks, S., 1998. Markov chain monte carlo method and its application. Journal of the royal statistical society: series D (the Statistician) 47, 69–100.

Cooley, D., Sain, S.R., 2010. Spatial hierarchical modeling of precipitation extremes from a regional climate model. Journal of agricultural, biological, and environmental statistics 15, 381–402.

Efeoglu, S., 2020. Propp-wilson_ising_model. `propp-wilson_Ising_Model`.

Eraker, B., 2001. Mcmc analysis of diffusion models with application to finance. Journal of Business & Economic Statistics 19, 177–191. URL: `https://doi.org/10.1198/073500101316970403`, doi:`10.1198/073500101316970403`, `arXiv:https://doi.org/10.1198/073500101316970403`.

Gallagher, K., Charvin, K., Nielsen, S., Sambridge, M., Stephenson, J., 2009. Markov chain monte carlo (mcmc) sampling methods to determine optimal models, model resolution and model choice for earth science problems. Marine and Petroleum Geology 26, 525–535. URL: `https://www.sciencedirect.com/science/article/pii/S0264817209000075`, doi:`https://doi.org/10.1016/j.marpetgeo.2009.01.003`. thematic Set on Basin Modeling Perspectives.

Goodman, J., Weare, J., 2010. Ensemble samplers with affine invariance. Communications in applied mathematics and computational science 5, 65–80.

Häggström, O., 2002. The Propp–Wilson algorithm. Cambridge University Press. London Mathematical Society Student Texts, p. 76–83.

Koike, T., Minami, M., 2019. Estimation of risk contributions with mcmc. Quantitative Finance 19, 1579–1597. URL: `https://doi.org/10.1080/14697688.2019.1588469`, doi:`10.1080/14697688.2019.1588469`, arXiv:https://doi.org/10.1080/14697688.2019.1588469.

Miazhynskaia, T., Dorffner, G., 2006. A comparison of bayesian model selection based on mcmc with an application to garch-type models. Statistical Papers 47, 525–549.

Mukhopadhyay, S., Bhattacharya, S., 2013. An improved bayesian semiparametric model for palaeoclimate reconstruction: Cross-validation based model assessment. arXiv: Applications URL: https://api.semanticscholar.org/CorpusID:88512924.

Pasha, I., 2020. Mcmc: A (very) beginnner's guide. https://prappleizer.github.io/Tutorials/MCMC/MCMC_Tutorial.html.

Propp, J.G., Wilson, D.B., 1996. Exact sampling with coupled markov chains and applications to statistical mechanics. Random Structures & Algorithms 9, 223–252.

Ross, K., 2022. Bayesian reasoning and methods. https://bookdown.org/kevin_davisross/bayesian-reasoning-and-methods/mcmc.html.

Wahiduzzaman, M., Yeasmin, A., Luo, J., Quadir, D.A., van Amstel, A., Cheung, K.K.W., Yuan, C., 2021. Markov chain monte carlo simulation and regression approach guided by el niño–southern oscillation to model the tropical cyclone occurrence over the bay of bengal. Climate Dynamics 56, 2693 – 2713. URL: https://api.semanticscholar.org/CorpusID:231745425.