Investigating the Efficiency of the Metropolis Algorithm

Tianye Dou, Aidan Li, Liyan Wang*
Department of Statistical Sciences
University of Toronto

Supervised by Jeffrey Rosenthal

May 3, 2024

Abstract

For Metropolis algorithms, an asymptotic acceptance rate of around 0.234 is optimal in the high-dimensional limit under certain restrictive conditions. However, the practical relevance of this figure is uncertain due to the unrealistic conditions underlying its derivation. In this study, we scrutinize the necessity of these assumptions. We conduct experiments across various scenarios to examine how robust the 0.234 rule is for maximising efficiency using Expected Squared Jumping Distance (ESJD), including Gaussian distributions with independent and non-independent components, different proposal distributions, multimodal distributions, and with parallel tempering's own optimal scaling theory. We find that the 0.234 acceptance rate generally maximises ESJD in many cases, but in some scenarios this is not empirically true even if theory says it should hold. Furthermore, maximising ESJD may not imply empirically good samples in multimodal or even unimodal distributions if the state space is not sufficiently explored. Lastly, experiments on parallel tempering show the idealized 0.234 spacing of inverse temperatures may not maximise ESJD for multimodal distributions even in some cases where the theory says it should.

*(alphabetical order)
We provide an open-source repository for our experiments: `https://github.com/aidanmrli/montecarlo`

Contents

1 Introduction

Markov Chain Monte Carlo (MCMC) methods, such as the Metropolis algorithm, are used to draw samples from complex high-dimensional probability distributions. They enjoy strong theoretical asymptotic guarantees of accuracy and flexibility to handle multimodal distributions. Metropolis algorithms converge to the target distribution in stationarity, but in practice, we would like to examine an algorithm's efficiency to ensure it performs reasonably well in finite time. Previous literature [4][8] shows that the acceptance rate of the algorithm is central to the algorithm's efficiency by dictating the balance between exploration of the state space and exploitation of high-density areas of the target distribution and that an asymptotic acceptance rate of around 0.234 is optimal in the high-dimensional limit under certain restrictive conditions. However, the validity of this figure hinges on strong assumptions that may not hold in practical scenarios, prompting scrutiny into its applicability in real-world contexts.

Our research examines how necessary these assumptions are for the theoretical result to remain relevant: where can we relax assumptions and still have the optimal acceptance rate of approximately 0.234, and where can we not do this? We thoroughly dissect various aspects of the Metropolis algorithm and experiment with them, and in doing so, we aim to show empirically where the theoretical ideal value can still align with more realistic scenarios.

As this has been a learning experience for us too, we write this paper with plenty of explanations for a possible beginner, and hope that it can be a gentle introduction to the optimal scaling problem in MCMC. We begin our paper with necessary background knowledge in Section 2. Next, we proceed to describe experiments with a Gaussian target distribution in Section 3, experiments with different proposal distributions in Section 4, and experiments with multimodal distributions in Section 5. Lastly, we introduce the parallel tempering method and its own optimal scaling and acceptance rate theory in Section 6, and provide experiments for parallel tempering on multimodal distributions.

2 Background

We first begin by motivating the problems that Monte Carlo tries to solve, then introduce the Metropolis algorithm and the Optimal Scaling problem. We then discuss measures to evaluate the efficiency of MCMC algorithms, such as the optimal acceptance rate and the expected squared jumping distance.

2.1 Monte Carlo Sampling

Monte Carlo methods are computational techniques that make use of random numbers. We use Monte Carlo methods to solve one or both of the following problems:

1. To generate samples $\{\mathbf{x}^{(r)}\}_{r=1}^{R}$ from a given probability distribution. This target distribution typically has a target density denoted $\pi(\mathbf{x})$.
2. To estimate expectations of functions under this distribution (using integration).

The target distribution might be complicated to sample from, especially if has *high dimensionality*. We cannot learn where $\pi(\mathbf{x})$ is big unless we evaluate $\pi(\mathbf{x})$ everywhere. To approximate the target density which is infeasible to directly compute, Monte Carlo methods typically use a simpler proposal density $Q(x)$ from which we can generate samples.

2.2 Metropolis-Hastings Algorithm

In high dimensions, it is difficult to create a single proposal density $Q(x)$ that is a good approximation to the target density $\pi(x)$. What we might do instead is construct a Markov chain that converges to the target distribution in the long-term after many runs, i.e. the Markov chain has a stationary distribution equal to the target distribution. When taking a next step in the Markov chain, we use a proposal density that depends on the current state $x^{(t)}$. The proposal density $Q(x'|x^{(t)})$ might be simple, such as a Gaussian centred on the current $x^{(t)}$, but can be any fixed density from which we can draw samples, and we examine examples with a non-Gaussian proposal distribution in Section 4. Unlike importance and rejection sampling, $Q(x'|x^{(t)})$ is not necessarily similar to $\pi(x)$. The Metropolis-Hastings algorithm is as follows:

1. Generate new state x' from the proposal density $Q(x'|x^{(t)})$.
2. Accept the new state with **acceptance probability** $a = \min\left(1, \frac{\pi(x')}{\pi(x^{(t)})}\frac{Q(x^{(t)}|x')}{Q(x'|x^{(t)})}\right)$.
3. If we accept the new state, $x^{(t+1)} = x'$.
4. If we reject the new state, $x^{(t+1)} = x^{(t)}$.

As $t \to \infty$, the probability distribution of $x^{(t)} \to P(x)$.

The Metropolis-Hastings algorithm is an example of a Markov chain Monte Carlo method (abbreviated MCMC). In contrast to rejection sampling, where the accepted points $\{x^{(r)}\}$ are independent samples from the desired distribution, Markov chain Monte Carlo methods use Markov chains in which a sequence of states $\{x^{(t)}\}$ is generated, i.e. each sample is dependent on the previous sample(s). Since successive samples are dependent, the Markov chain may have to be run for a considerable time to generate samples that are effectively independent samples from π.

Note: The Metropolis algorithm is a simpler case of the Metropolis-Hastings algorithm. The Metropolis algorithm assumes a symmetric random walk proposal $Q(x^{(t)}|x') = Q(x'|x^{(t)})$, thus the acceptance probability a simplifies to $a = \min\left(1, \frac{\pi(x')}{\pi(x^{(t)})}\right)$, i.e. just comparing the target density at the two points. For the rest of this paper, we use the Metropolis algorithm instead of the more complicated Metropolis-Hastings.

2.3 Optimal Scaling Overview

The Metropolis algorithm should eventually converge to the target distribution if we take an infinite number of steps, but this is not practical. We therefore have two issues:

1. Speed of convergence: we want to take fewer steps to get a reasonably good approximation.
2. Accuracy of the approximation: we want to explore all the modes of the target distribution.

The performance of the algorithm often hinges on the choice of proposal distribution, as illustrated by Figure 1. Different choices of the proposal distribution's "scaling", such as the standard deviation σ of the Gaussian distribution, lead to very different results. If the standard deviation σ is too small (the proposal density is too narrow), we usually accept the proposed steps, but the Markov chain won't explore the state space much and not all the modes of the target density $\pi(x)$ might be visited in a finite number of steps. On the other hand, if σ is too large, we usually reject the proposed steps and are very slow to make new exploration steps, so we take extremely long to converge to $\pi(x)$.

Therefore, the proposed standard deviation σ is crucial to the performance of the algorithm. Different Metropolis algorithms have shown to have different optimal choices of the proposal distribution variance σ^2, but overall, they are inversely proportional to the dimension d.

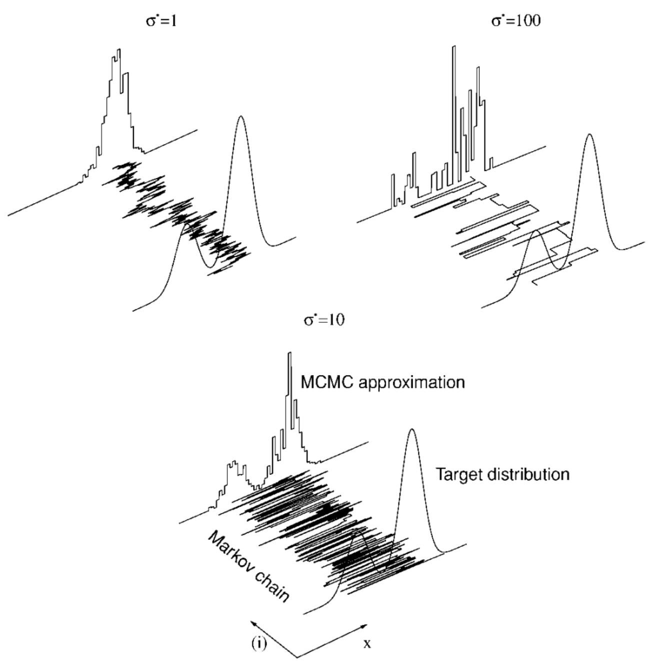


Figure 1: Approximations obtained using the MH algorithm with three Gaussian proposal distributions of different variances [1]. For each subfigure, the samples are the top-left histogram and the target distribution is the bottom-right curve. The Markov chain traceplot is between the samples and target distribution.

2.4 Evaluating the Efficiency of MCMC Algorithms

Having discussed the optimal scaling problem, it then makes sense to think about how to evaluate the efficiency of a MCMC algorithm. There are various notions of efficiency of a Markov chain, but in the high-dimensional limit $d \to \infty$ where the chain converges to a diffusion process, all efficiency measures are effectively equivalent [4][8][9]. A key result of these referenced papers is that, given a Metropolis algorithm with an increment proposal distribution $Q = N(0, \frac{\ell^2}{d} I_d)$ where $\ell > 0$ is a fixed scaling constant and I_d is the identity matrix, maximising the algorithm's *speed measure*, which is a function of the scaling constant ℓ, yields the most efficient asymptotic diffusion. Furthermore, the limiting speed has a clear relation to a much simpler quantity to estimate: the asymptotic acceptance rate of the proposed new states (moves) of the algorithm, defined as

$$a = \lim_{n \to \infty} \frac{\#\text{ accepted moves}}{n}.$$

Under certain restrictive conditions, the optimal asymptotic acceptance rate for a Random-Walk Metropolis algorithm has been proven [4][8] to be approximately 0.234. However, the proof requires very strong and possibly unrealistic assumptions, which means the 0.234 figure may not necessarily be optimal for certain cases where these assumptions are not satisfied. Our research examines how necessary these assumptions are for the theoretical result to still be relevant: where can we relax assumptions and still have the optimal acceptance rate be approximately 0.234, and where can we not do this?

2.4.1 Expected Squared Jumping Distance

For the experiments in this report, we largely lean on the *Expected Squared Jumping Distance* (ESJD) metric to define a measure of efficiency in finite dimensions. The expected squared jumping distance measures how far, in expectation, the MCMC chain moves in a simple iteration. For the standard random-walk Metropolis algorithm, we define this as

$$\mathbb{E}\left[||x^{(t+1)} - x^{(t)}||^2\right].$$

Maximising ESJD aligns with minimizing the first-order auto-correlation of the Markov chain and subsequently maximises efficiency if the higher-order auto-correlations are monotonically increasing relative to the first-order auto-correlation [5][10]. Under the restrictive conditions where the optimal acceptance rate is provably 0.234, this acceptance rate of 0.234 also maximises the ESJD of the algorithm [10].

3 Simulations with the Gaussian Distribution

In this section, we explore the target distribution with Gaussian densities using the standard Random Walk Metropolis Algorithm. We analyze the effectiveness of the 0.234 acceptance rate rule and the proposed variance of $2.38^2/d$ in these scenarios.

3.1 Multivariate Gaussian example with i.i.d. components

The main findings from this type of example are:

- The ESJD (efficiency) is maximised when the acceptance rate is approximately 0.234 for dimensions up to 100.
- Choosing the proposal variance of the proposal distribution where ESJD is maximised will make the proposal distribution converge to the target distribution, as demonstrated by a

well-mixing trace plot and a good convergence of the histogram for each coordinate. The optimal proposal variance is approximately equal to $2.38^2/d$.

- Different starting points will not make any difference in the optimal proposal variance and convergence performance.
- With scaling factors in the target distribution, it is efficient to use proposal distribution $Q(x) = N(0, \sigma^2 diag(C_1, ..., C_d))$, where $C_1, C_2, ...C_d$ are the scaling factors.

3.1.1 Without scaling factors

Assume dimension $d = 30$ and the number of iterations is 200,000. Roberts & Rosenthal (2001) [8] showed that in the Gaussian scenario, if the correlation structure is accurately determined, then appropriately scaling the proposal in each direction to match the target scaling optimises the efficiency of the algorithm. Let's consider target distribution $\pi(x) = \prod_{i=1}^{d} f(x_i)$, define $f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{x^2}{2}\right)$, and consider a proposal distribution $Q(x) = N(0, \sigma^2 I_d)$. The convergence and mixing of the algorithm vary with different proposal variances. Taking $\sigma^2 = 0.001$ as an example, the corresponding trace plot exhibits poor mixing (see Figure 2). By exploring a range of proposal variances, we can generate a plot that illustrates the relationship between ESJD and the acceptance rate (See Figure 3). Each point on this plot corresponds to a specific proposal variance, along with its associated ESJD and acceptance rate.

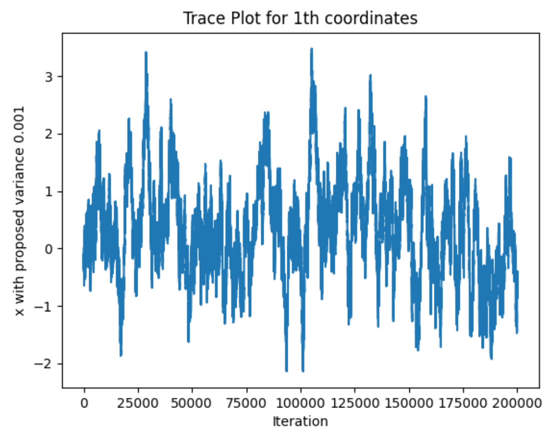


Figure 2: Trace plot of 1st coordinate with $\sigma^2 = 0.001$ for i.i.d. multivariate Gaussian distribution

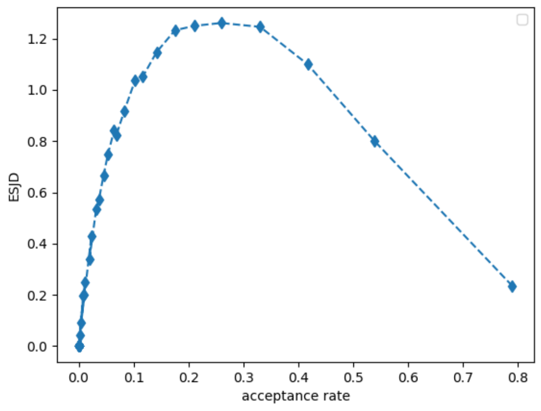


Figure 3: ESJD for Metropolis Algorithm as a function of acceptance rate

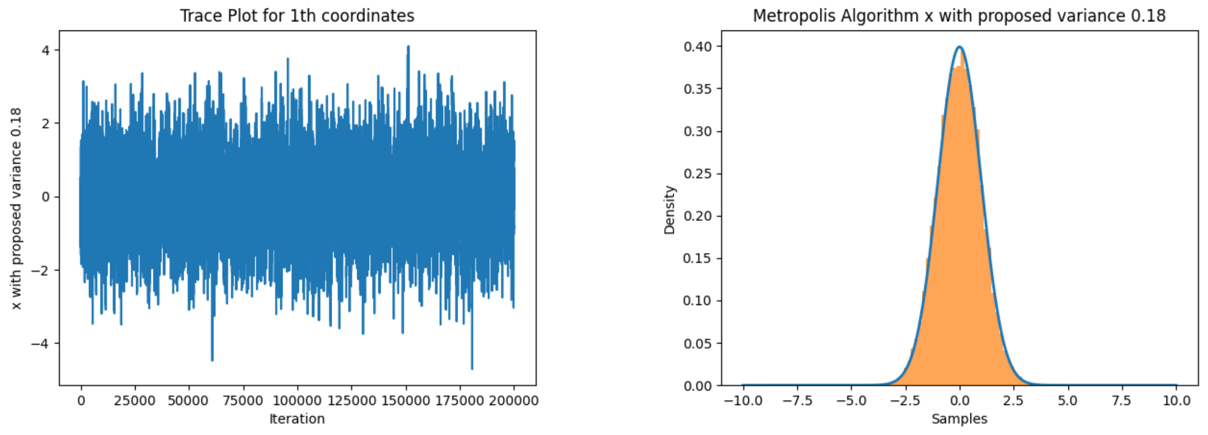


Figure 4: Trace Plot and histogram of 1st coordinate with $\sigma^2 = 0.18$ for i.i.d. multivariate Gaussian

Upon examining the plot, the optimal variance that maximises ESJD is found to be approximately 0.18, correlating to an acceptance rate of about 0.231. Remarkably, this closely aligns with the theoretically optimal acceptance rate of 0.234, with $0.18 \approx 2.38^2/30$. Furthermore, setting the proposed variance to 0.18 yields encouraging results: the trace plot for the first coordinate (See Figure 4 left panel) indicates good mixing of samples, and the histogram confirms convergence to the target distribution (See Figure 4 right panel). This example illustrates that for a Gaussian distribution with i.i.d. components, the Metropolis algorithm optimized by ESJD tends to produce well-fitting models when starting from 0.

Next, we consider a variation where the target distribution's single-component density $f(x)$ is a standard double exponential distribution defined by $f(x) = \frac{1}{2}\exp(-|x|)$. We maintain the same dimension $d = 30$ and number of iterations at 200,000, with the proposal distribution $Q(x) = N(0, \sigma^2 I_d)$. Exploring a range of proposed variances, we find the acceptance rate that maximises ESJD approaches 0.3 (See Figure 1 in Appendix). This rate does not closely align with the "magic" acceptance rate of 0.234. Yet, with the corresponding proposed variance of $\sigma^2 = 0.17$, the trace plot shows effective mixing, and the histogram aligns with the target distribution after the burn-in phase. Hence, it can be inferred that aiming for an acceptance rate exactly or near 0.234 is not critically necessary for efficiency; rates between 0.15 and 0.5 achieve approximately 80% efficiency [8].

The initial conditions are also examined. We assume initial points are randomly selected from -15 to 15 and conduct parallel sampling due to its low computational cost. The experiments (See Figure 5) suggest that different starting points do not significantly influence the acceptance rate that maximises ESJD, which consistently hovers around 0.23. This indicates that the initial position does not substantially affect the convergence speed or efficacy of the Metropolis Algorithm in homogeneous multivariate Gaussian scenarios.

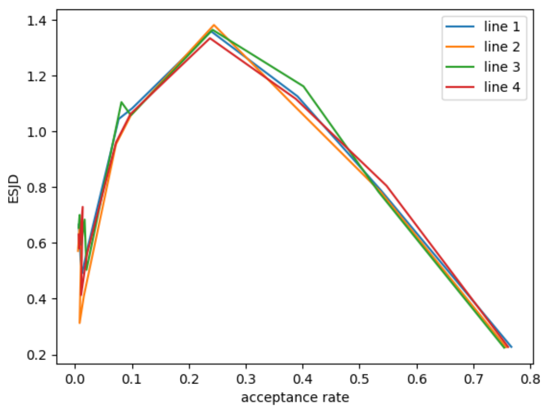


Figure 5: Different starting points for Gaussian distribution with i.i.d. components

Additionally, we extended our investigation to understand how the algorithm performs in higher dimensions, up to 100. Under the same assumption of the target distribution as described above, we execute simulations with varying σ^2 values and plot the ESJD as a function of the acceptance rate (see Figure 6) across dimensions $d = 10, 30, 50, 100$. Remarkably, the maximal ESJD was consistently achieved at an acceptance rate of approximately 0.234 across all examined dimensions, and as the dimension increases, the curves converge to the theoretical curve $v(l)$:

$$v(l) = l^2 a(l) \tag{1}$$

where $a(l)$ represents the asymptotic acceptance rate. These plots also demonstrated an increase in ESJD with dimensionality, and the optimal acceptance rate approached the theoretical asymptote 0.234, corroborating the findings reported by Bédard (2008) [3]. This suggests that for target distributions with Gaussian i.i.d. components and a proposal distribution $Q(x) = N(0, \sigma^2 I_d)$, where σ^2 is approximately equal to $2.38^2/d$, maintaining a maximised ESJD and an acceptance rate close to 0.234 is advantageous regardless of dimension. These findings highlight the robustness and broad applicability of this acceptance rate, confirming its effectiveness for different starting points and across various high dimensions.

3.1.2 Scaling factors

This section explores the concept of heterogeneity of scale within components for more general experiments. We assume the target density, denoted by $\pi(x)$, can be expressed as $\pi(x) =$

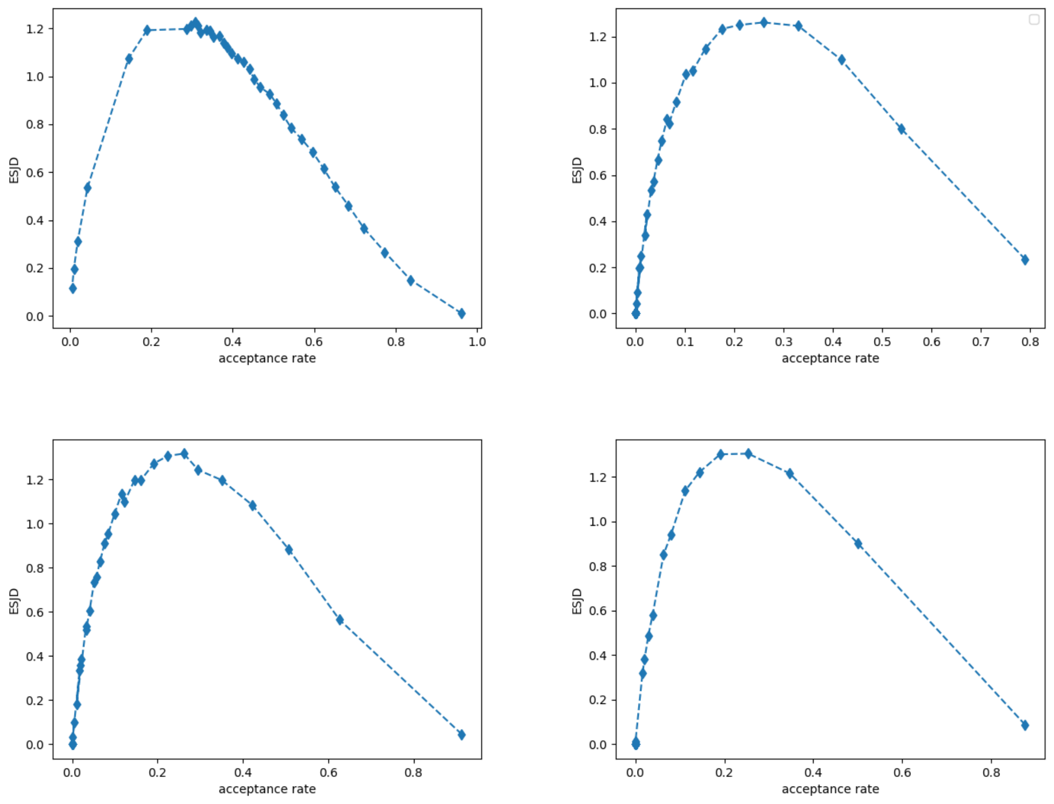


Figure 6: ESJD of Gaussian examples with i.i.d. components as a function of acceptance rate, with dimension = 10 (top left), 30 (top right), 50 (bottom left), 100 (bottom right)

$\prod_{i=1}^{d} C_i f(C_i x_i)$. Here, f is standard Gaussian, $E(C_i) = 1$ for all components ($i = 1$ to d) and the constants C_i have finite variance.

Known Constants: Let's assume we know all the constants C_i (where $i \in \{1, 2, ..., d\}$ and $d = 30$). To ensure their expected value is 1, we can randomly choose these constants from a uniform distribution between 0.2 and 1.8. Our proposal distribution is denoted by $Q(x) = N(0, \sigma^2 I_{30})$.

In our analysis, we explore the trace plot and histogram for both the first and third coordinates to evaluate the sampling performance. The scaling factors for these coordinates are 1 and 0.206 respectively. Referring to the relationship between the ESJD and acceptance rate depicted in Figure 7 (left panel), we observe that the ESJD for the first coordinate peaks at 0.036 with an acceptance rate of approximately 0.252. Through further experimentation, we identify an optimal proposal variance of $\sigma^2 = 0.165$.

Employing this optimal variance, the trace plot for the first coordinate displays satisfactory

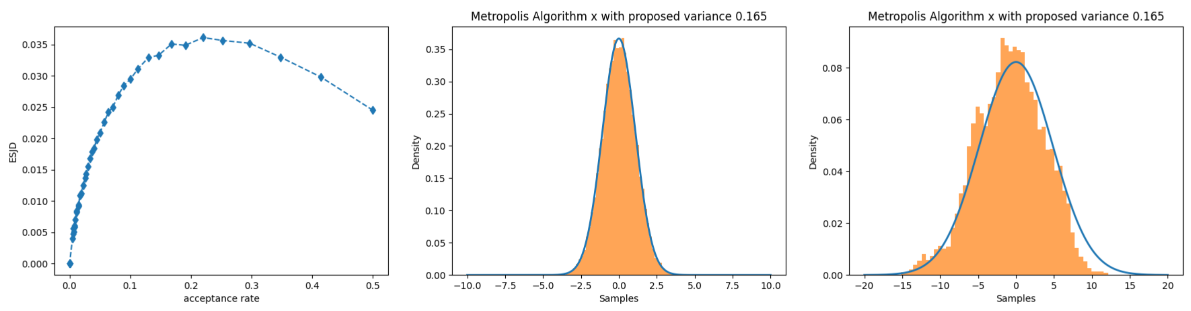
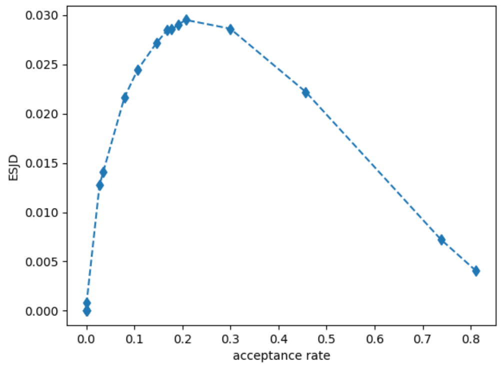


Figure 7: Known scaling factors with homogeneous proposal distribution. Left: relation of ESJD and acceptance rate for the first coordinate. Middle: Histogram of 1st coordinate with $\sigma^2 = 0.165$. Right: Histogram of 3rd coordinate with $\sigma^2 = 0.165$.

mixing behavior, and the histogram (Figure 7, middle panel) aligns well with the target distribution. However, when the same variance is applied to the third coordinate, the histogram performance deteriorates significantly (Figure 7, right panel). This discrepancy arises because the target distribution $\pi(x)$ assigns different scaling factors to different coordinates. By setting the proposal distribution as a standard multivariate Gaussian with a covariance matrix of $\sigma^2 I_d$, the proposal variance uniformly scales each coordinate, potentially leading to convergence issues for coordinates with scaling factors that differ markedly from the proposal variance. In this scenario, the scaling factor for the first coordinate is exactly 1, facilitating effective sampling under a homogeneous proposal distribution. Conversely, for the third coordinate, where the scaling factor deviates significantly from the proposal distribution's implicit scaling of 1, the sampling efficacy is compromised.

Moreover, Figure 7 (left panel) illustrates that the maximum ESJD is approximately 0.037. Subsequent experiments reveal that the maximum ESJD value for the first coordinate in Section 3.1.1 is approximately 0.044. According to Theorem 5 from Roberts & Rosenthal (2001) [8], the efficiency of scaling factors with a homogeneous proposal distribution follows a similar pattern, peaking when the acceptance rate is around 0.234. However, this is scaled down by a factor of about 0.8, closely matching $C_1^2/b \approx 0.83$, where $b = E(C_i^2) = 1.21$. This application of Theorem 5 demonstrates that at least when $C_1 = 1$, the efficiency of the first coordinate with a homogeneous proposal function in a scaling context is reduced by the factor b compared to a non-scaling situation.

Given that the scaling factors are known constants, a logical approach would be to select a proposal distribution that closely mirrors the target distribution. This means the proposal distribution would scale each coordinate similarly to the target distribution. Let us denote this as the



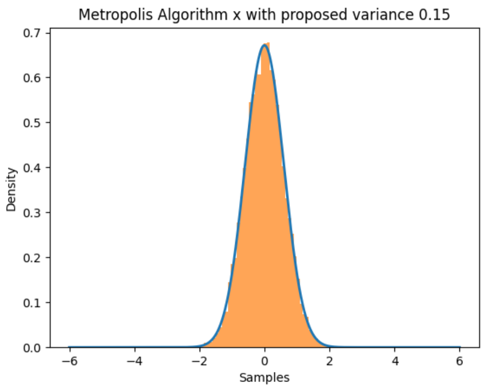


Figure 8: Known scaling factors with proposal distribution $Q(x) = N(0, \sigma^2 diag(C_1, ..., C_{30})$. Left: relation of ESJD and acceptance rate. Right: Histogram of 13th coordinate with $\sigma^2 = 0.15$.

inhomogeneous proposal distribution $Q(x)$, characterized by a multivariate Gaussian distribution $N(0, \sigma^2 \text{diag}(C_1, ..., C_{30}))$.

Adopting this adjusted proposal distribution and following the same evaluative procedure as before, we find that the efficiency of the Metropolis algorithm is again maximised at an acceptance rate of approximately 0.234. With the corresponding optimal proposed variance ($\sigma^2 = 0.15$), the trace plot for a randomly selected coordinate (in this instance, the 13th) shows good mixing, and the histogram aligns closely with the target distribution (as depicted in Figure 8). Similarly, the histograms and trace plots for the other coordinates indicate satisfactory convergence, with the exception of the third coordinate. The improved performance of the proposal distribution in this scenario is clear: by matching the scaling degree of the proposal to that of the target distribution for each coordinate, the scaling factors effectively neutralize each other. This results in a sampling performance that closely resembles the scenario in Section 3.1.1, where no scaling is applied.

Unknown constants: When the constants C_i are unknown, the specific proposal distribution $Q(x) = N(0, \sigma^2 \text{diag}(C_1, ..., C_{30}))$ becomes infeasible. From our previous analysis and the plots in Figure 7 (left panel) and Figure 8 (left panel), we observe that the maximal ESJD of homogenous proposal distribution (which is approximately 0.036) is higher than the ESJD of inhomogeneous proposal distribution (which is 0.031). This implies that the efficiency of a homogeneous proposal distribution is notably higher than that of an inhomogeneous proposal algorithm. It is important to note that this conclusion is based on the generated plots, but it may not intuitively make sense, and we will discuss this in detail in the discussion (Section 3.3). Employing a standard multivariate Gaussian proposal distribution remains a viable option, providing a robust fallback when detailed scale information is unavailable.

3.2 Multivariate Gaussian example with non-i.i.d. components

The key results we found from this type of example are:

- For non-i.i.d. components of the multivariate Gaussian examples, merely aiming to achieve an optimal acceptance rate of approximately 0.234 is insufficient.
- Introducing a form of proposal distribution could be beneficial to sample non-i.i.d. multivariate target distributions, such as $Q(x) = \mathcal{N}(0, 2.38^2/d \cdot \Sigma)$, where Σ is the covariance matrix of the target distribution, or more practically, $Q(x) = \mathcal{N}(0, 2.38^2/d \cdot \Sigma_n)$, where Σ_n $= COV(X_1, X_2, ...X_{n-1})$.
- For target density $\pi(x) = \prod_{i=1}^{d} f_i(x_i)$ in high dimensions, if f_i converges to f, and $Q(x) = \mathcal{N}(0, \sigma^2 I_d)$, then the 0.234 rule and $\sigma^2 = 2.38^2/d$ are efficient for sampling.

3.2.1 Non-independent components

In the context of the Metropolis algorithm, the primary focus has traditionally been on achieving an acceptance rate approximation of 0.234. However, it is crucial to recognize that the optimal acceptance rate alone does not fully gauge the algorithm's effectiveness for a target distribution. To illustrate this, let's consider a scenario with non-independent components in the covariance matrix with dimension 30. For the target distribution

$$\pi(x) = \mathcal{N}(0, \Sigma_\pi)$$

where Σ_π must be symmetric and positive-definite, we define $\Sigma = M^T \cdot M$. Here, M is a 30×30 matrix consisting of independent and identically distributed (i.i.d.) $\mathcal{N}(0, 2)$ components, and M^T represents the transpose of M. The proposal distribution is still assumed to be i.i.d., with $Q(x) = \mathcal{N}(0, \sigma^2 I_{30})$. When running the algorithm with σ^2 set to 0.04, the acceptance rate is approximately 0.23469, yet the trace plot exhibits inefficient mixing and convergence (See Figure 9). Thus, $\sigma^2 = 0.04$ is not a suitable estimate for the proposal distribution in this case, even though the acceptance rate is close to the optimum. This insight extends across dimensions (e.g., $d = 10$ or $d = 100$), demonstrating that an acceptance rate close to 0.234 does not necessarily guarantee effective sampling outcomes. This underscores the necessity of considering additional characteristics and diagnostics beyond the acceptance rate.

The theorem formalized by Roberts & Rosenthal (2001) [8] suggests that the optimal proposal variance is given by $Q(x) = \mathcal{N}(0, 2.38^2/d \cdot \Sigma_\pi)$. In this specific example, $Q(x) = \mathcal{N}(0, 0.1888 \cdot \Sigma_\pi)$. Under this assumption, the final trace plot of the first coordinate exhibits efficient mixing (shown

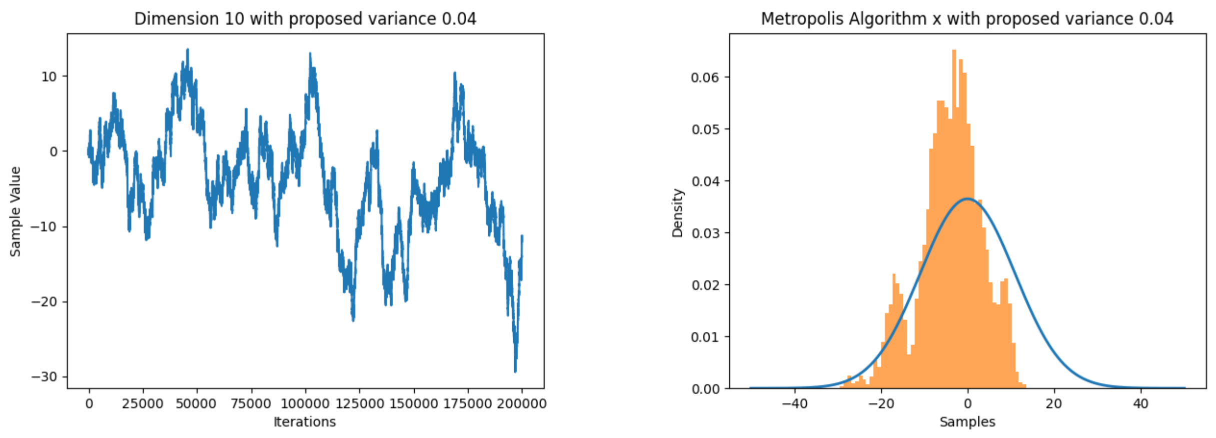


Figure 9: Trace Plot and Histogram of 10th coordinate non-iid multivariate function with $\sigma^2 = 0.04$, having proposal distribution $Q(x) = \mathcal{N}(0, \sigma^2 I_{30})$

in Figure 10), and the acceptance rate is around 0.244, which is close enough to the optimal acceptance rate of 0.234. From this example, it becomes evident that while a good convergence sampling often has an acceptance rate close to 0.234, not every sampling with an acceptance rate near 0.234 exhibits efficient mixing and convergence. Thus, the acceptance rate alone cannot be the sole criterion for judgment. The fundamental goals are to assess whether the trace plot is reasonably mixing well and whether the histogram of samples converges to the target distribution for most of the coordinates. Additionally, the proposed variance adjustment $Q(x) = \mathcal{N}(0, 2.38^2/d \cdot \Sigma_\pi)$ proves to be a valuable tool in high-dimensional Gaussian distributions.

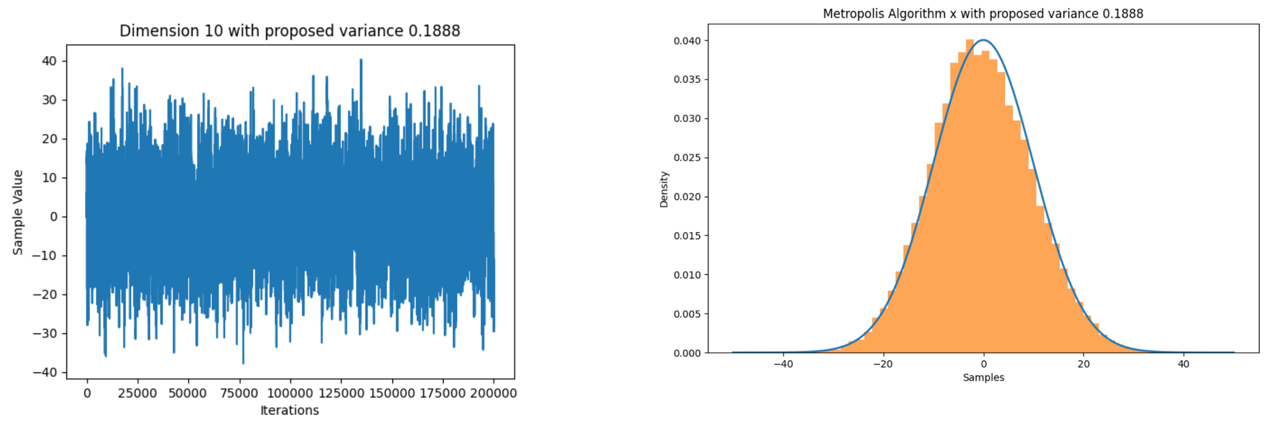


Figure 10: Trace Plot and Histogram of 10th coordinate non-iid multivariate function with $\sigma^2 = 0.1888$, having proposal distribution $Q(x) = \mathcal{N}(0, 2.38^2/d \cdot \Sigma_\pi)$

However, defining the proposal distribution as $Q(x) = \mathcal{N}(0, 2.38^2/d \cdot \Sigma_\pi)$ is sometimes restrictive, and the necessary assumption for this is that the covariance matrix of target distribution is

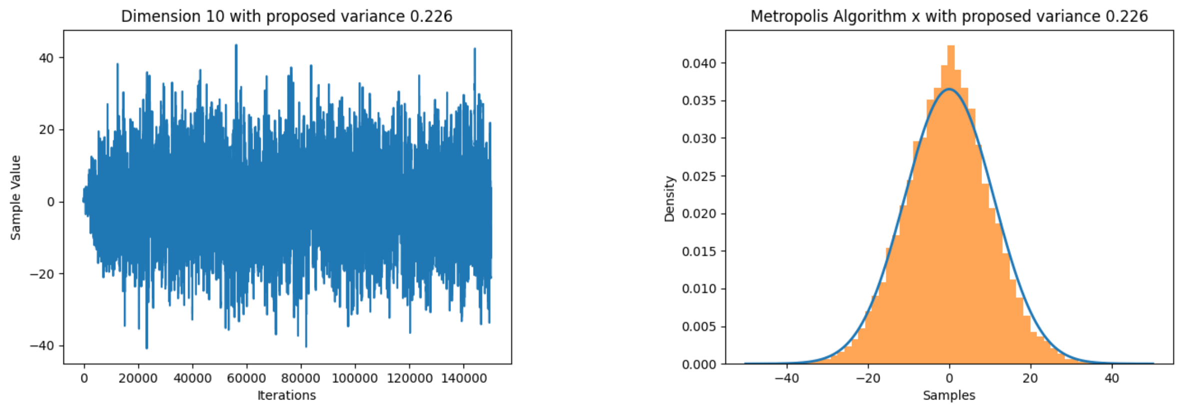


Figure 11: Trace plot and histogram of 10th coordinate non-iid multivariate function with σ^2 =0.226, having proposal distribution $Q(x) = \mathcal{N}(0, 0.226 \cdot \Sigma_n)$

known or easy to get. In a more realistic situation, Σ_π would not be known in advance, and it would be difficult to estimate it or its unbiased estimator.

Haario et al.[2001] [6] modified the random-walk Metropolis algorithm by computing the covariance matrix of the Gaussian proposal distribution from a finite number of all previous states. Sometimes, to increase the computation speed under a huge number of iterations, the covariance matrix can be updated every 50 or 100 iterations. Note that it is not too complicated to update the covariance matrix for each iteration by applying a simple recursion.

The adaptive Metropolis Algorithm enhances the proposal distribution by dynamically updating the covariance matrix Σ_n to approximate Σ_π. This matrix Σ_n is derived from the covariance of the visited vectors $X_0, \ldots, X_{n-1}$ up to the current iteration X_n, making it a robust estimate if these X_i are representative of π. The proposal distribution Q is defined as:

$$Q(x) = \begin{cases} N(0, \sigma^2 I_d) & \text{if } \textit{iteration} < 20 \\ N(0, \sigma^2 \Sigma_n) + \epsilon N(0, \sigma^2 I_d) & \text{if } \textit{iteration} \geq 20 \end{cases}$$

Here, ϵ is a small constant introduced to prevent Q's covariance matrix from becoming singular, though it can be set to zero in non-multimodal contexts as per Harrio et al. (2001) [6]. Analysis of ESJD against the acceptance rate for various σ^2 values reveals that ESJD is maximized at an acceptance rate of 0.253 with a corresponding $\sigma^2 = 0.226$. This rate suggests good performance as evidenced by the trace plot and histogram of the first coordinate (See Figure 11). The chosen proposal variance of 0.226 is equal to $2.6^2/d$ and does not closely approximate $2.38^2/d = 0.1888$, but both of them work well. The reason behind the difference between 0.226 and the theoretical 0.1888 proposal variance might be that the dimension 30 is too small to show the theoretical pattern of the optimal proposal variance. This conclusion highlights that, although convergence time,

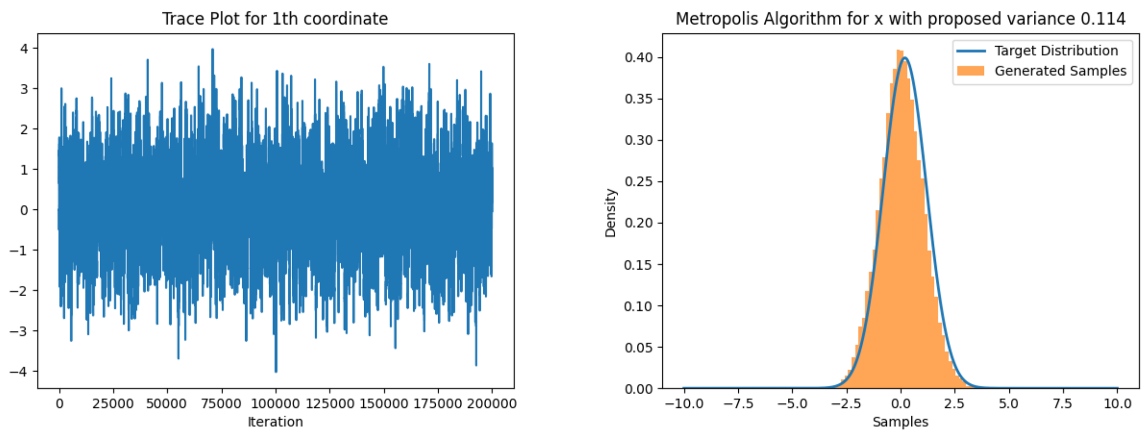


Figure 12: Trace plot and histogram of 1st coordinate

speed, and efficiency may vary when applying different proposal distributions, aiming for $2.38^2/d$ in the proposal variance typically approaches a maximal efficiency within the given proposal distribution framework.

3.2.2 non-identically distributed components

Assuming $d = 50$, it is insightful to consider scenarios where different components of the target distribution are not identically distributed. According to Roberts, Gelman, and Gilks (1997) [4], consider a target density defined as $\pi(x) = \prod_{i=1}^{d} f_i(x_i)$, with each $f_i(x) = N(\frac{1}{5i}, 1)$. As i increases, f_i converges to a standard Gaussian distribution $f(x) = N(0, 1)$. Following the established procedure, the proposal density is defined as $Q(x) = N(0, \sigma^2 I_{50})$. Analysis of the relationship between ESJD and acceptance rate, depicted in Figure 2 in Appendix, indicates that efficiency is maximized at an acceptance rate of approximately 0.24. Experimentation reveals that the corresponding variance is 0.114, which intriguingly matches the "optimal" proposed variance for i.i.d. components of the target density calculated as $2.38^2/d \approx 0.114$. Utilizing this proposal variance $\sigma^2 = 0.114$, the trace plots and histogram of the first coordinate (See Figure 12) demonstrate effective convergence of the proposed density to the target density. Additional figures (Figure 5 and Figure 6 in Appendix) illustrate similar convergence performance for other coordinates.

This evidence suggests that Theorem 1, traditionally limited to i.i.d. cases, may also apply to certain non-i.i.d. special scenarios. This extension of the theorem's applicability demonstrates the robustness of the $2.38^2/d$ rule in optimizing proposal variances, even in more complex distribution frameworks.

3.3 Discussion

The Metropolis algorithm demonstrated efficient convergence and sampling performance for the multivariate Gaussian target distribution. Convergence was assessed through visual inspection of trace plots, which indicated stable behavior and convergence to the stationary distribution. The samples obtained from the Metropolis algorithm exhibited characteristics consistent with the properties of the target distribution and demonstrated favourable performance in high-dimensional settings. In the analysis above, we primarily focused on dimension 30, and some cases of dimension 100. Although in dimension 30, the data results and comparisons we obtained align closely with what the theory predicts regarding the 0.234 acceptance rule, using higher dimensions would be more persuasive. In Section 3.1.2, concerning known scaling factors, we utilized the proposal distribution $Q(x) = N(0, \sigma^2 \text{diag}(C_1, \ldots, C_{30}))$. Since both the target and proposal distributions scale each coordinate identically, the scaling effects between the two distributions effectively cancel each other out. This theoretically results in sampling performance similar to Section 3.1.1, where no scaling was applied. Additionally, under an inhomogeneous proposal distribution, the algorithm does not need to consider every scaling factor of each coordinate, as the scaling factors have already been cancelled out and we can consider them to have the same scaling factor of 1. In contrast, a homogeneous proposal distribution must account for every scaling factor of the target distribution, leading to larger errors and complexity, thereby increasing convergence time and reducing efficiency. Thus, the efficiency of a homogeneous proposal distribution is notably lower than that of an inhomogeneous proposal algorithm. In practice, if the scaling factors are unknown, it may be reasonable to estimate the scaling factors or their unbiased estimators to increase the efficiency of the algorithm. Nonetheless, our experiments indicate variance in the “optimal” proposal variance, suggesting the need for further refinement and exploration to align experimental outcomes more closely with theoretical predictions, particularly regarding optimizing proposal variance under varied scaling conditions.

4 Proposal Distributions

Although the Gaussian distribution is the most widely used proposal distribution, there are limitations of the Gaussian distribution as the chosen proposal distribution.

- The Gaussian distribution is characterized by rapidly decaying tails, which can pose challenges when exploring target distributions with heavy tails or multiple modes. In such scenarios, the Gaussian proposal distribution may fail to adequately explore the full range of the target distribution, potentially causing the Markov chain to get stuck near a single mode in multi-modal distributions.
- The Gaussian distribution spans all real numbers. If the distribution we are sampling from is only defined on positive values, the Gaussian proposal may generate values where the target density equals zero. These proposed values are promptly rejected, preventing the Markov chain from effectively exploring the parameter space because the Markov chain does not move from the current position.

In this section, we will investigate different proposal distributions for the Metropolis Algorithm.

4.1 Double Exponential distribution as proposal distribution (i.i.d.)

The double exponential distribution is similar to the Gaussian distribution. It is continuous and symmetric, but it has wider tails than the Gaussian distribution and it is more peaked. The double exponential distribution has two parameters: a location parameter μ, which defines its mean, and a scale parameter b, which defines the width of the distribution.

In this case, since we have a symmetric proposal distribution, the acceptance ratio can be simplified to $a = min(1, \frac{\pi(y)}{\pi(x)})$. Suppose we have independent target components, we let π be the product form: $\pi(x) = \prod_{i=1}^{d} f(x_i)$, and the proposal $Q(X)$ is the double exponential distribution with location parameter 0 and scale parameter $\sigma^2 I_d$.

4.1.1 Example

Suppose $f(x_i)$ is the standard Gaussian density and the target distribution has product form $\pi(x) = \prod_{i=1}^{d} f(x_i)$, and the proposal distribution is the double exponential distribution with its scale parameter $\sigma^2 I_d$, with σ to be chosen and location parameter 0.

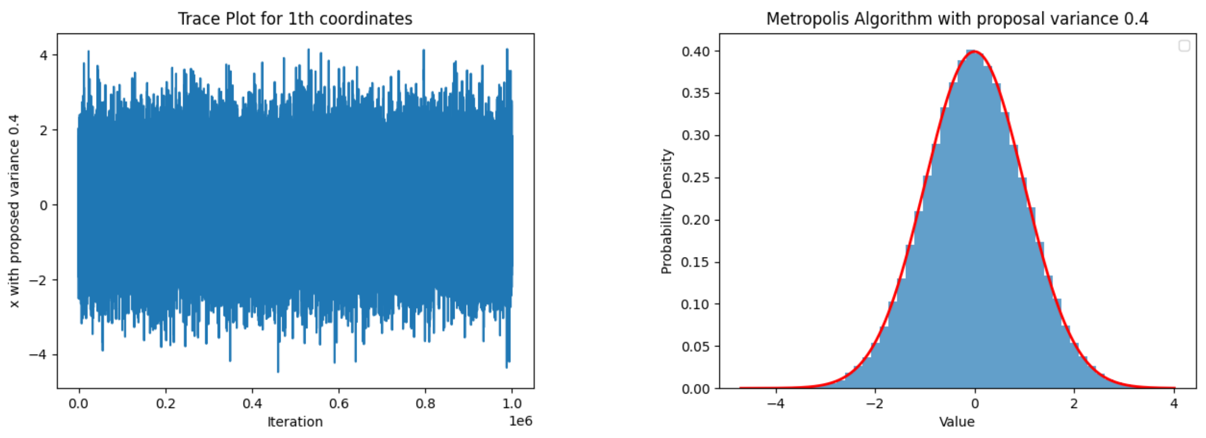


Figure 13: Trace Plot and Histogram of the 1st coordinate (Double Exponential proposal distribution) with $\sigma^2 = 0.4$

We set the dimension to 20 and the number of iterations to 1,000,000. The scale parameter of the proposal distribution is $\sigma^2 I_{20}$, where I_{20} is the 20-dimensional identity matrix. From the graph depicting the relationship between ESJD and proposal variance, we observe that when the proposal variance is approximately 0.4, the ESJD reaches its maximum. Plotting the trace plot of the first coordinate when the proposal variance is 0.4, we find that it mixes well. Moreover, the plot of the histogram of the first coordinate indicates a convergence trend towards the target distribution (Figure 13). This observation underscores the important role of proposal variance in optimizing the efficiency of the Metropolis algorithm. Moreover, at this critical variance setting, the acceptance rate stabilizes around 0.23, closely aligning with a theoretical value of 0.234.

Expanding our investigation into higher dimensions, up to 100, we find ESJD peaks as the acceptance rate approaches 0.234 (see Figure 14).

In this example, we also explore the influence of different starting points on the Metropolis Algorithm. Five different starting points are generated. Each component of the starting points is generated independently from a uniform distribution between -10 and 10. Plotting the ESJD as a function of the acceptance rate, the graph indicates that they have similar behavior. The ESJD is maximized when the acceptance rate is close to 0.234 (Figure 15). Thus, choosing different initial points does not affect the choice of the optimal proposal variance, indicating the robustness of the Metropolis Algorithm to the choice of initial conditions.

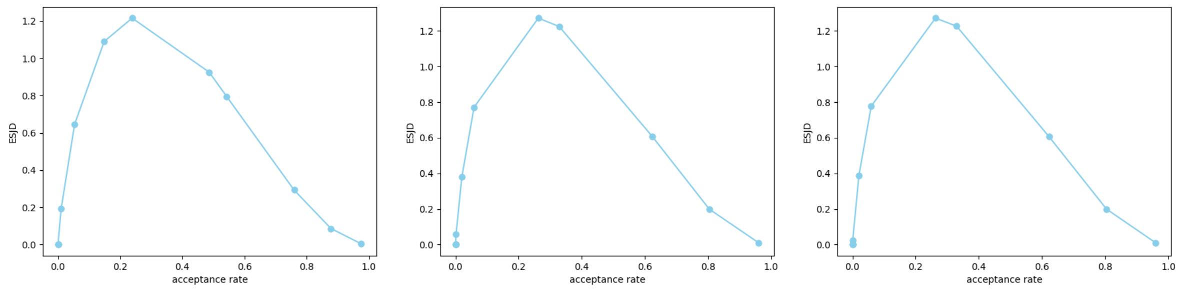


Figure 14: ESJD as a function of acceptance rate (Double Exponential proposal distribution), with dimension = 20 (left), 50 (middle), 100 (right)

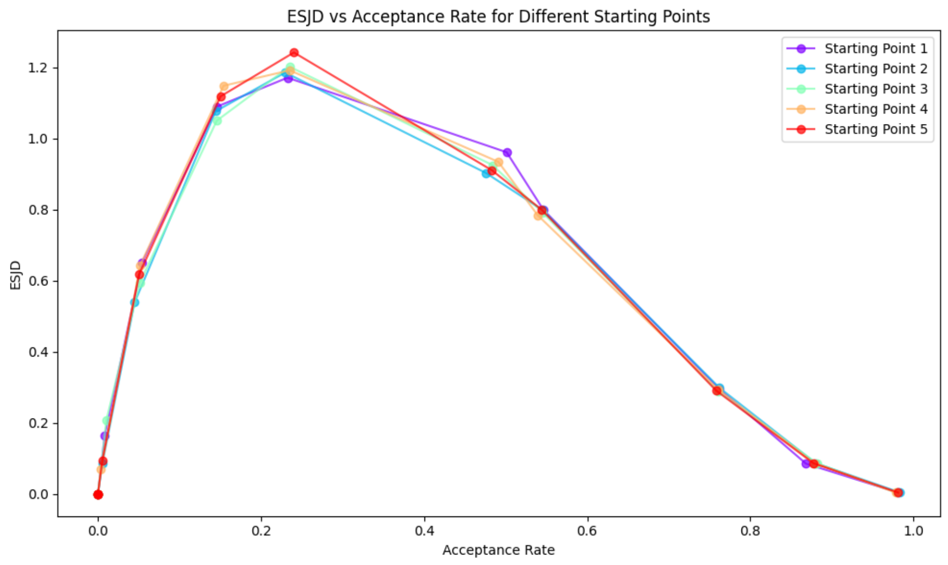


Figure 15: Different starting points with Double Exponential Proposal Distribution

4.2 Uniform distribution as proposal distribution (i.i.d.)

Uniform distributions are probability distributions with equally likely outcomes. In a continuous uniform distribution, outcomes are continuous and infinite. Additionally, uniform distributions are symmetric.

In this case, we have a symmetric proposal distribution, so the acceptance ratio can be simplified to $\alpha = min(1, \frac{\pi(y)}{\pi(x)})$. Suppose we have independent target components. We let π be the product form: $\pi(x) = \prod_{i=1}^{d} f(x_i)$, and set the proposal as the uniform distribution.

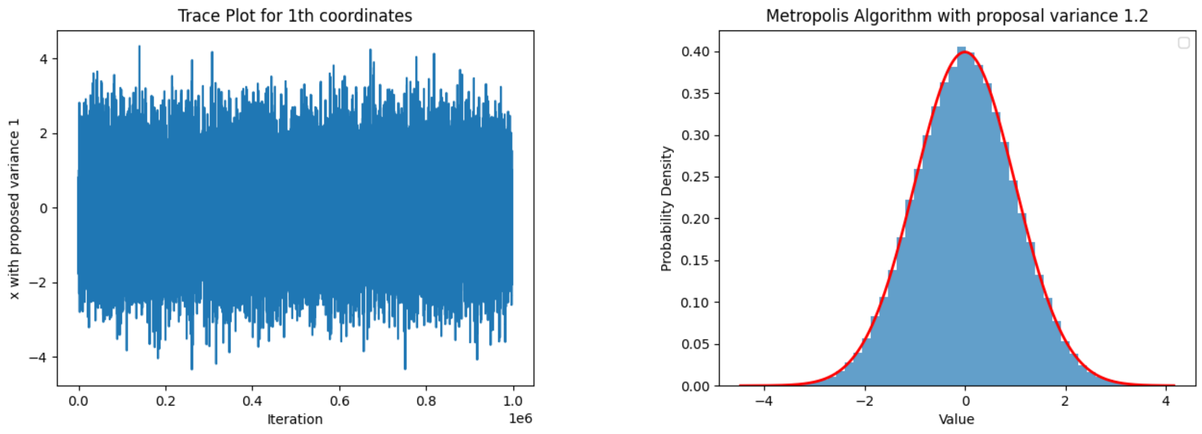


Figure 16: Trace Plot and Histogram of the 1st coordinate (Uniform proposal distribution) with $\sigma^2 = 1.2$

4.2.1 Example

In this scenario, we consider a target distribution comprised of independent components. We let π be the product form: $\pi(x) = \prod_{i=1}^{d} f(x_i)$, where each $f(x_i)$ represents the standard Gaussian distribution. Given $X_n = x$, the proposal distribution is the uniform distribution within the interval $[x - \frac{stepsize}{2}, x + \frac{stepsize}{2}]$.

With the dimension set to 50 and the number of iterations at 1,000,000, ESJD attains its maximum when the proposal value is around 1.2. Notably, ESJD diminishes to zero for proposal variances exceeding 4, showing that the chain is not moving at all between successive iterations. Fixing the proposal variance at 1.2, its acceptance rate is close to 0.234. However, for a proposal variance larger than 4, its acceptance rate is 0. Visualization through trace plots and histograms of the first coordinate suggests that the proposal variance is effectively optimized (Figure 16).

Expanding our investigation, we explore the algorithm's behavior for dimensions of 20 and 100, plotting ESJD as a function of the acceptance rate with 1,000,000 iterations. Our findings indicate that ESJD reaches its maximum when the acceptance rate approximates 0.234, regardless of the dimensionality (Figure 17).

In this example, we also investigate whether different starting points affect the selection of the optimal proposal variance. By generating five distinct starting points, with each component sampled independently from a uniform distribution spanning -10 to 10, we find consistent maximization of ESJD when the acceptance rate converges towards 0.234 (Figure 18).

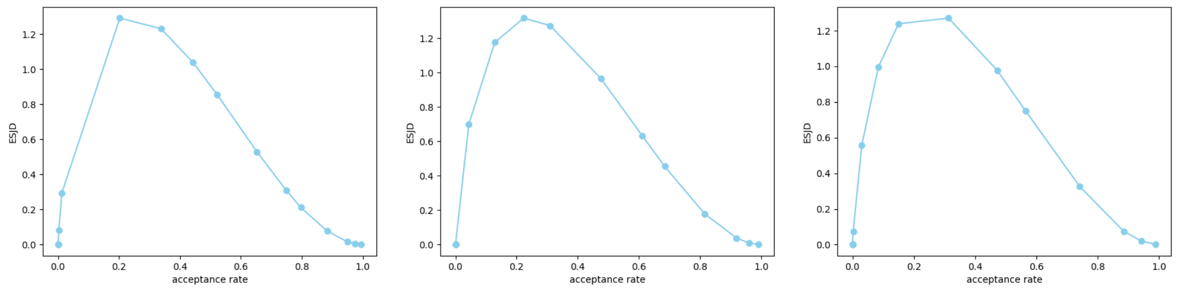


Figure 17: ESJD as a function of acceptance rate (Uniform Proposal distribution), with dimension = 20 (left), 50 (middle), 100 (right)

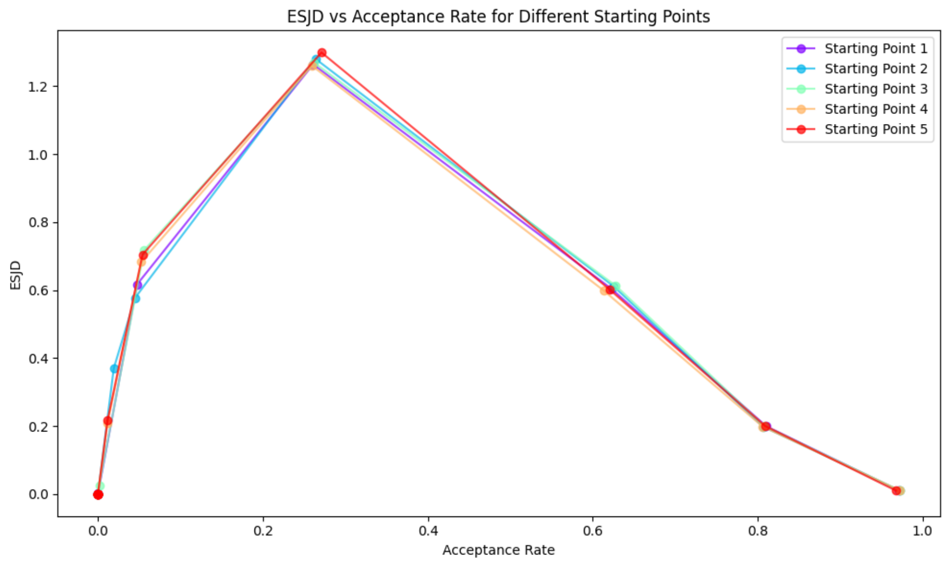


Figure 18: Different starting points with Uniform Proposal Distribution

4.3 Summary of Experimenting with Different Proposal Distributions (i.i.d.)

- ESJD peaks when the acceptance rate is approximately 0.234, for dimensions up to 100.
- Choosing the proposal variance that maximises the ESJD is efficient, which is indicated by the well-mixed trace plots and the convergent histograms for each coordinate.
- Variation in starting points does not have an impact on the choice of the optimal proposal variance.

4.4 Inhomogeneous Target Distribution with Different Proposal Distributions

4.4.1 Inhomogeneous Gaussian Target distribution

According to Atchadé, Roberts & Rosenthal (2011)[2], we can modify previous examples to a non-i.i.d. case where $\pi(x) = \prod_{i=1}^{d} f(x_i)$ with $f_i = N(0, i^2)$. In this case, we have the dimension equal to 20 and run the experiment for 100,000 iterations.

Firstly, when the proposal distribution adheres to a Gaussian distribution $Q(x) = N(0, \sigma^2 I_{20})$, our analysis reveals that the ESJD reaches its peak for a proposal variance of 6. Although the trace plot of the first coordinate indicates effective mixing, the trace plot of the 20th coordinate suggests that the proposal variance is small (Figure 19).

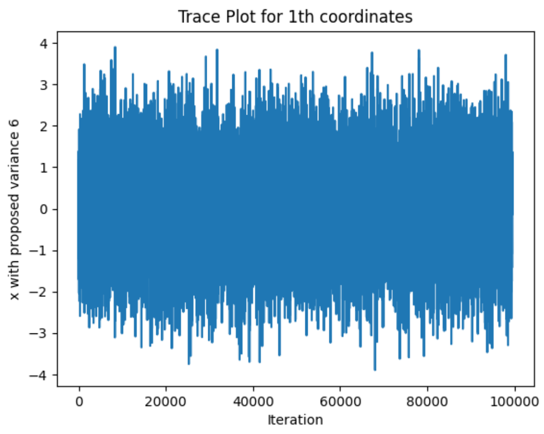
Then, we change the proposal $Q(X)$ to be the Double Exponential distribution with location parameter 0 and scale parameter $\sigma^2 I_{20}$. The proposal variance 2 maximises the ESJD. However, the trace plot of the 19th coordinate suggests that the proposal variance is small (Figure 7 in Appendix).

At last, we choose the proposal distribution to be in the form of $Q(x) = N(0, \sigma^2 diag(1^2, 2^2, ...20^2)$. When the proposal is 0.237, ESJD is maximized. The trace plots of all coordinates mix well and the histogram of all coordinates show a good convergence to the target distribution (Figure 20). Moreover, the acceptance rate is close to 0.234 as ESJD is maximised (Figure 8 in Appendix).

4.4.2 Inhomogeneous Double Exponential Target distribution

In this non-i.i.d. case, we have $\pi(x) = \prod_{i=1}^{d} f(x_i)$ with $f_i = \frac{e^{-|x/i|}}{2i}$. In this case, the dimension is 20 and the number of iterations is 100,000.

We consider a Gaussian proposal distribution $Q(x) = N(0, \sigma^2 diag(1^2, 2^2, ...20^2)$. The ESJD is maximized when the proposal variance is 0.37. By examining the trace plot and the histogram of each coordinate, we find that they show an effective convergence (Figure 21). Furthermore, the observed maximization of ESJD aligns with an acceptance rate close to 0.234 (Figure 9 in Appendix).



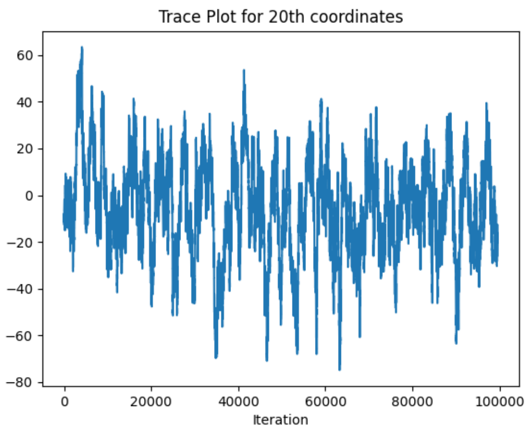
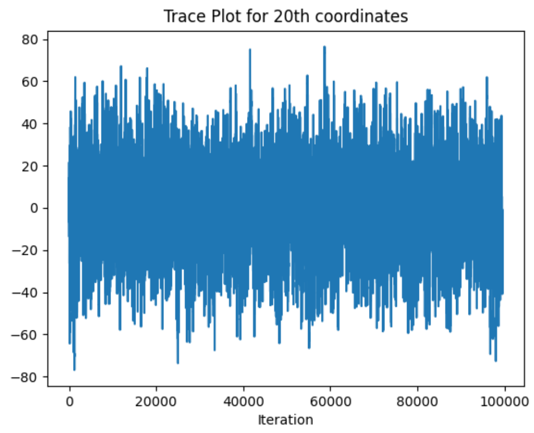


Figure 19: Trace plot with proposal variance 6: 1st coordinate (left), 20th coordinate (right)



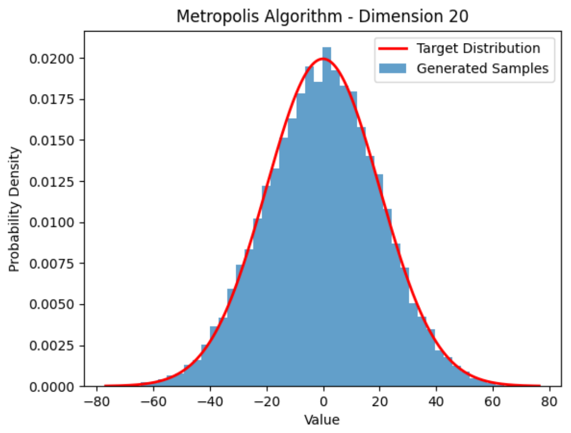


Figure 20: Trace Plot and Histogram of the 20th coordinate ($Q(x) = N(0, \sigma^2 diag(1^2, 2^2, ...20^2))$ with proposal variance 0.237

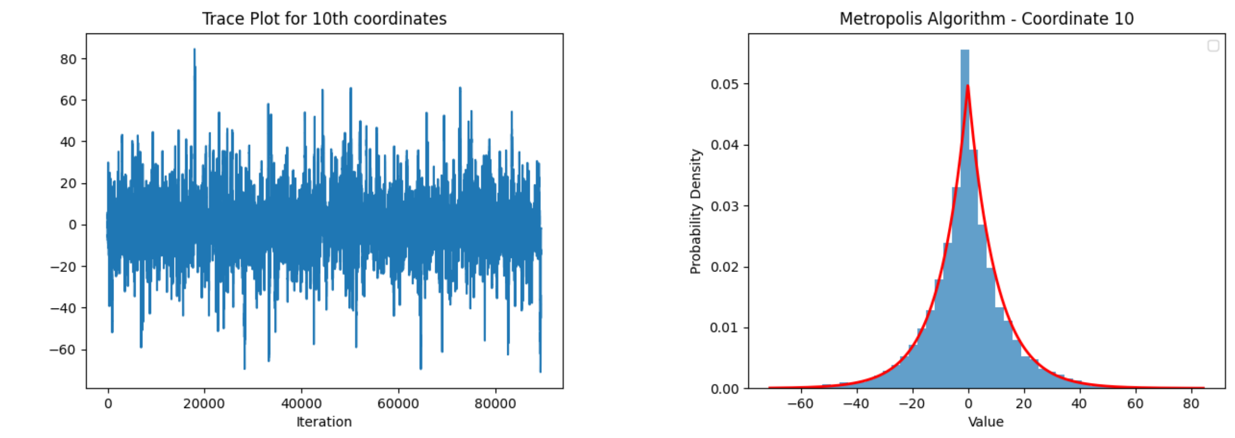


Figure 21: Trace Plot and Histogram of the 10th coordinate ($Q(x) = N(0, \sigma^2 diag(1^2, 2^2, ...20^2))$ with proposal variance 0.37

4.5 Discussion

We investigate different proposal distributions using the Metropolis Algorithm. We consider different proposal distributions, namely the Double Exponential distribution and the Uniform distribution.

A critical aspect of the experiments is the optimization of proposal variance to maximise the efficiency of the Metropolis Algorithm. We observe that the ESJD is maximized when the acceptance rate is approximately 0.234. Moreover, the experiments demonstrate that the choice of initial points does not significantly affect the selection of the optimal proposal variance, indicating the stability and reliability of the algorithm across different starting points. The extension of the analysis to inhomogeneous target distribution offers insights into adapting proposal distributions to complex and diverse sampling scenarios. By considering the scale of the target distribution, proposal distributions can be tailored to improve exploration and convergence in challenging environments.

While the experiments provide valuable insights, it is important to acknowledge its limitations and potential areas for future research. The focus on unimodal and symmetric target distributions may not fully capture the complexity of real-world settings. Future studies can explore the effectiveness of proposal distributions in more diverse and realistic scenarios.

5 RWM on Multimodal Target Distributions

In this section, we investigate target distributions with multimodal densities using the standard Random Walk Metropolis Algorithm. We examine how useful the 0.234 rule is in these cases where the theory is not necessarily applicable.

5.1 The "Rough Carpet" Distributions

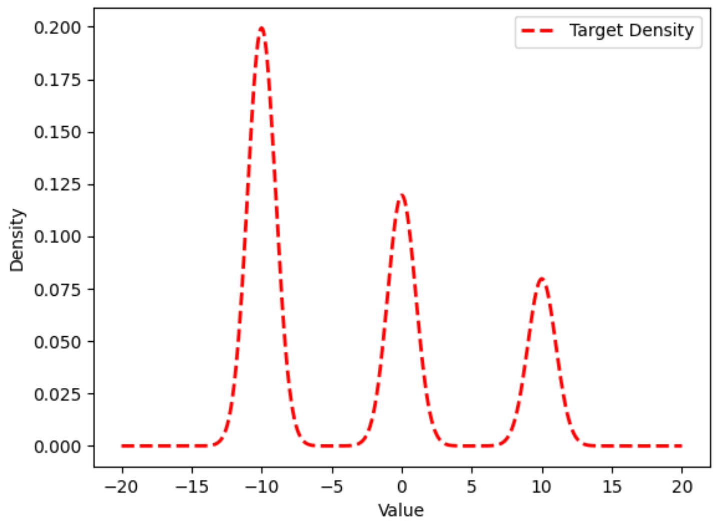
First, we conduct experiments on families of distributions with the following characteristics:

1. We examine a multimodal distribution with i.i.d. components of the form $\pi = \prod_{i=1}^{n} f(x_i)$.
2. We examine a multimodal distribution of the form $\pi = \prod_{i=1}^{n} C_i f(C_i x_i)$, with heterogeneous component-wise scaling factors C_i, where $C_i > 0$ and $\mathbb{E}(C_i) = 1$ and $(C_i) < \infty$.

In both cases, our single component density f is a one-dimensional density with three modes. We define our density:

$$f(x) = 0.5N(x|m_1, 1) + 0.3N(x|m_2, 1) + 0.2N(x|m_3, 1) \tag{2}$$

where $m_1, m_2, m_3 \in \mathbb{R}$ and $N(x|m_i, 1)$ indicates the density of the univariate Gaussian distribution at point x with mean m_i and variance 1.



This forms a "rough carpet"-like appearance for the target distribution over the many dimensions (see the visualization of the first component in Figure 5.1). Since we have 3^d modes, the density values at each mode may not be that high. We ran various experiments in dimensions $d \in \{5, 20, 30, 50\}$.

5.1.1 Adherence to the 0.234 theory

In this experiment, we aimed to investigate how well the 0.234 optimal acceptance rate theory would hold for this unconventional multimodal target distribution. The theory should apply to the target distributions described in Section 5.1 since they are products of i.i.d. single-component densities, but is this the case in practice?

For this experiment, we ran many simulations with dimension $d = 20$ where each simulation was a random-walk Metropolis algorithm with 40 different variance values from $0.01^2/d$ to $4.0^2/d$ (and subsequently, a different acceptance rate). Each Metropolis algorithm ran for 100,000 iterations. Furthermore, we run each algorithm instance over 5 different seeds and average the results to reduce the effects of randomness caused by a particular seed. Overall, this experiment had 200 simulations which each ran for 100,000 iterations.

We set the modes to be reasonably close so that the standard random-walk Metropolis algorithm would be able to jump between modes somewhat frequently. In this experiment, we use the single-component density $f(x) = 0.5N(x \mid -5, 1) + 0.3N(x \mid 0, 1) + 0.2N(x \mid 5, 1)$.

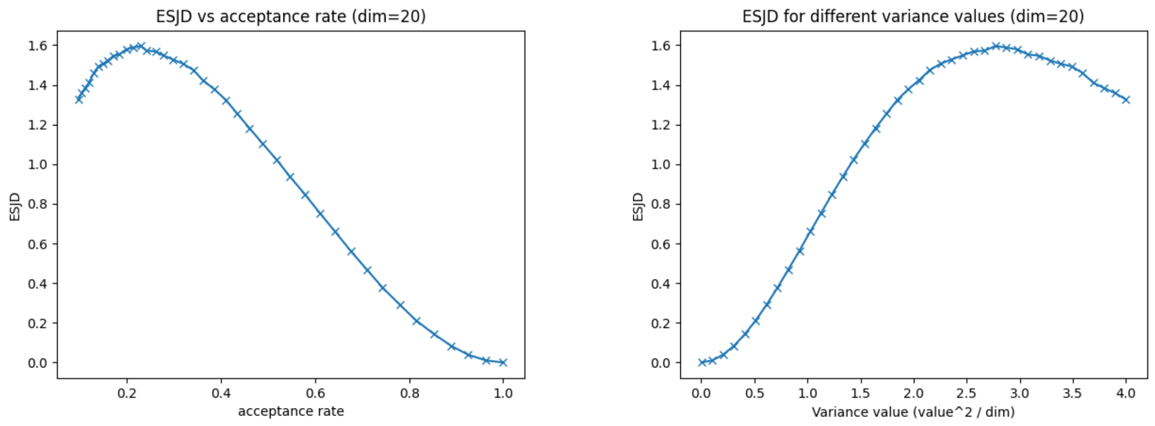


Figure 22: Non-scaled distribution ($d = 20$): ESJD against acceptance rate and variance.

We report our results for the non-scaled target distribution (Section 5.1 point 1) in Figure 22. We find that ESJD is maximised at an acceptance rate of approximately 0.234. This is consistent with the theory as our target density is still a product of a single-dimensional density in the d components. However, the proposal variance corresponding to that 0.234 acceptance rate is not $2.38^2/d$; rather, the result is that $2.872^2/d$ optimizes the acceptance rate.

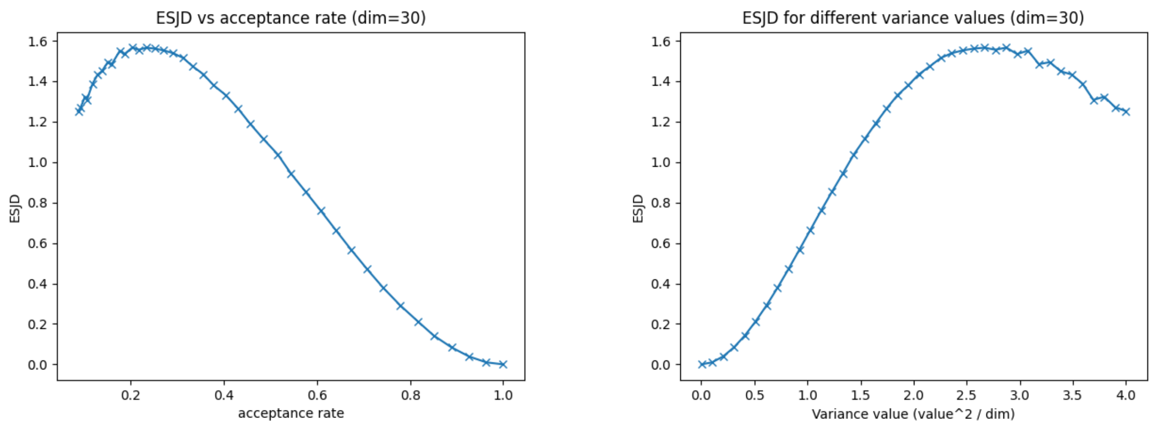


Figure 23: Non-scaled distribution ($d = 30$): ESJD against acceptance rate and variance.

To test whether the optimal variance not being $2.38^2/d$ was simply due to the dimension $d = 20$ being too low, we repeated this experiment for $d = 30$ and found that there was no noticeable change, with the variance $2.872^2/d$ maximising the ESJD yet again. The results are included in Figure 23.

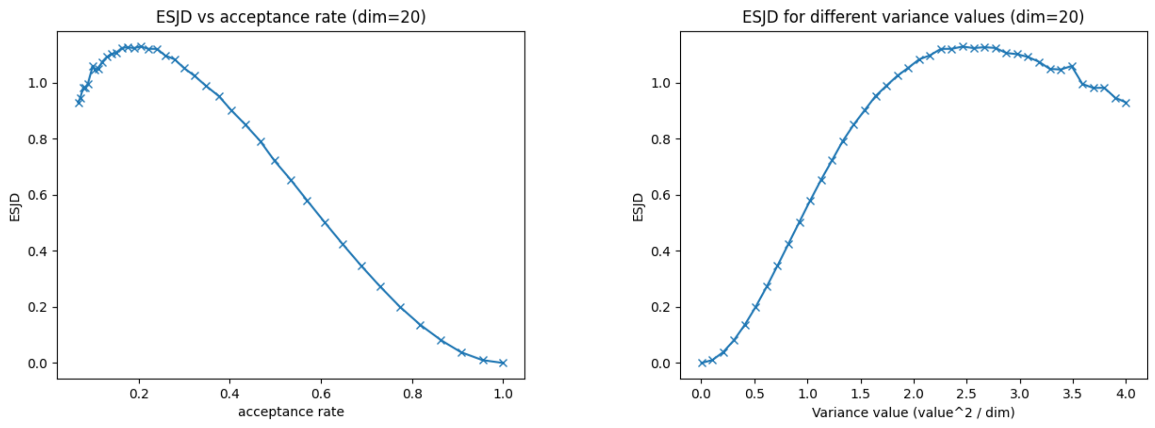


Figure 24: Scaled distribution ($d = 20$): ESJD against acceptance rate and variance.

We report our results for the scaled target distribution (Section 5.1 point 2) in Figure 24. Here, we find that the results are no longer consistent with the theory. We find that ESJD is maximised at an acceptance rate of approximately 0.205 instead of 0.234. This might be because of the random scaling factors; although the individual C_i's are sampled from Uniform$[0, 2]$ which has mean 1, the sampled mean of these scaling factors may not necessarily have mean 1. Furthermore, the proposal variance that maximises ESJD is approximately $2.462^2/d$.

5.1.2 Impact of mode separation on proposal distribution

We ran various experiments where we visually inspected what would happen when we changed how far apart the individual modes were. The aim was to learn to what extent a Gaussian proposal distribution would continue to be effective as we try target distributions with modes that are further apart.

For each experiment, we ran a single instance of the random-walk Metropolis algorithm with a fixed proposal variance for up to 10 million iterations. We tried different proposal variances for a fixed target mode distance, and if the results were satisfactory, we moved the modes of the target further apart. We ran these experiments in dimension $d = 5, 20, 50$.

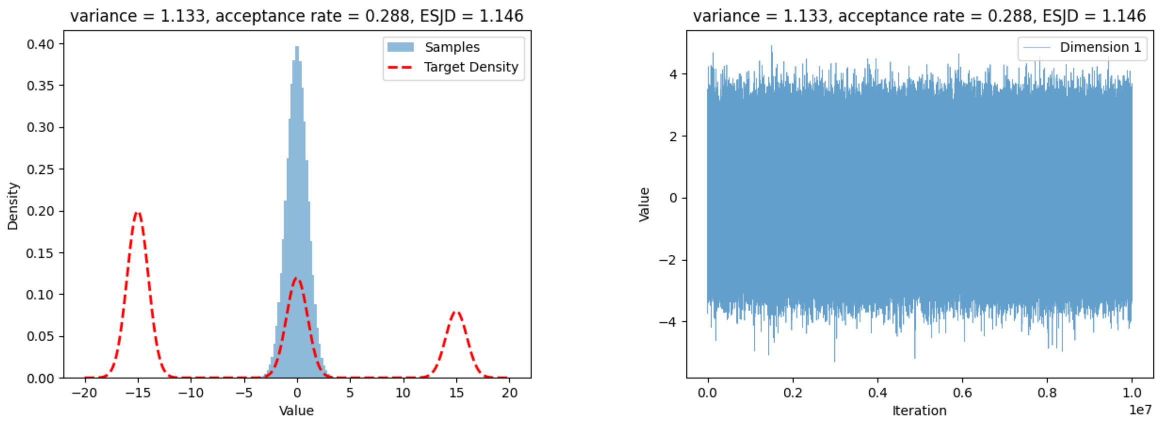


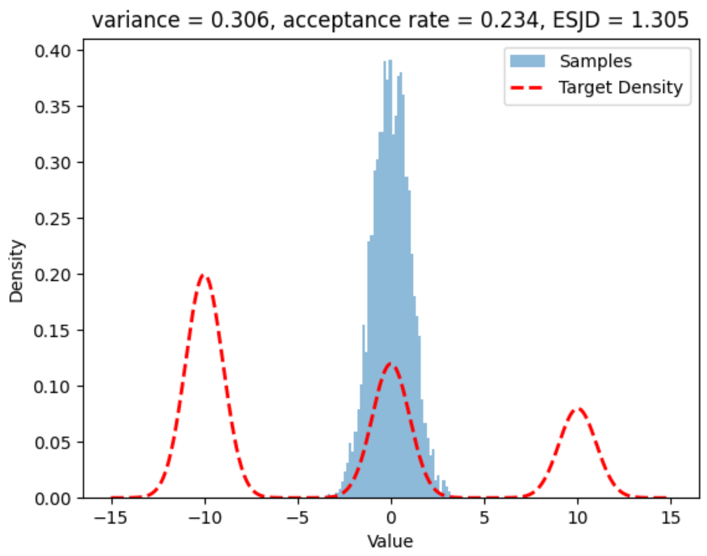
Figure 25: Histogram and traceplot of the first component over 10 million iterations.

Since the optimal proposal variance is inversely proportional to the dimension d, we find that as we increase the dimension, the values of the individual component means m_1, m_2, m_3 need to be closer together for a RWM with a Gaussian proposal to continue being effective. When we have widely-separated modes, a RWM with the Gaussian proposal distribution struggles to explore the other modes regardless of the proposal scaling and the dimension of the target distribution. For example, in Figure 25, we ran our experiments for up to 10 million iterations and did not escape the central mode once. We find this result somewhat surprising: although the modes might be far apart, and Section 7.2 of Roberts and Rosenthal (2001) says the convergence time should be exponential in d, we have 10^7 iterations in this experiment.

Perhaps the reason that a Gaussian proposal density does not seem very useful in this case is that making the proposal variance larger makes the proposal distribution more flat. In other words, making the proposal distribution wider does not make the probability of hitting the far away target modes substantially higher; it just makes the distribution more uniform. Since there is no one area with a higher probability, picking a proposed state that coincides with the higher-density areas of the target distribution is almost like playing high-dimensional Battleship.

5.1.3 Challenges with acceptance rate and ESJD

That being said, optimizing the acceptance rate no longer guarantees an "optimal" sampling. We might have an acceptance rate of 0.234 and a maximised ESJD while still being stuck in a single mode, as this figure 5.1.3 shows (experiment at $d = 20$, 1 million iterations). We also often oversample from a smaller mode while failing to draw samples from a larger mode, or completely miss a mode altogether even with a large number of iterations. This may be due to the inherent limitations of the random walk algorithm.



In theory, ESJD should reward jumping between modes more, and a higher ESJD is supposedly indicative of taking more between-mode jumps as opposed to jumps within the same mode. That being said, it suffers from the same problem that the acceptance rate does, albeit to a smaller degree; it can still be high without jumping between modes that often if it takes large jumps within the mode frequently. It might be worth investigating a different metric in the future, such as the asymptotic variance over many runs.

5.2 A Tale of Three Mixtures

The above "rough carpet" distributions are interesting, but notice that they still follow the general product form $\pi = \prod_{i=1}^{n} f(x_i)$. What if we try a distribution that does not have this i.i.d. assumption? The theory does not technically apply to this since the proof does not hold.

In this section, we examine a mixture of three Gaussians; a multimodal distribution with just three modes regardless of the number of dimensions. The general form of this density is

$$\pi(x) = \frac{1}{3}N(x|m_1, \Sigma_1) + \frac{1}{3}N(x|m_2, \Sigma_2) + \frac{1}{3}N(x|m_3, \Sigma_3) \tag{3}$$

where N is a multivariate Gaussian distribution with mean $m_i \in \mathbb{R}^d$ and covariance matrices Σ_i. For these experiments, we set $m_1 = (c, 0, 0, \ldots, 0), m_2 = (0, 0, \ldots, 0), m_3 = (-c, 0, 0, \ldots, 0)$ for some constant $c \in \mathbb{R}$, i.e. just varying the first component of the means and setting the other components of the means to 0. We start our Markov chain at $(0, 0, \ldots, 0)$.

5.2.1 Optimal acceptance rates to maximise ESJD

As mentioned, this distribution violates many assumptions required of the 0.234 acceptance rate theory. Is an acceptance rate of 0.234 still good to aim for if the theory does not apply? This experiment aims to answer this question.

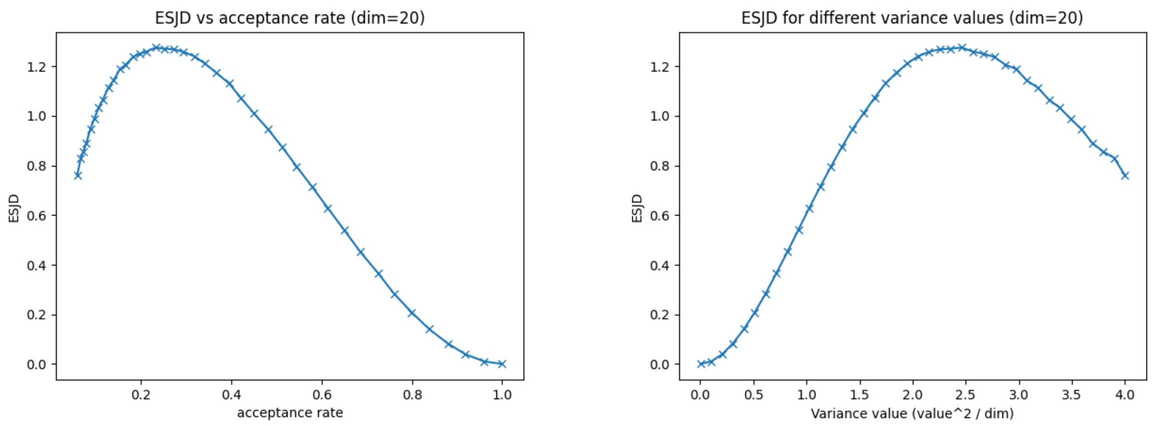


Figure 26: ESJD against acceptance rate and variance. ($d = 20$)

Our setup for this experiment is exactly the setup described in Section 5.1.1, but with a different target distribution. We conduct the experiment at dimensions $d = 20, 30$. We first conduct some preliminary experiments to choose the constant c in the first dimension to be some value such that it is not easy for the chain to jump between modes because the modes are reasonably far apart,

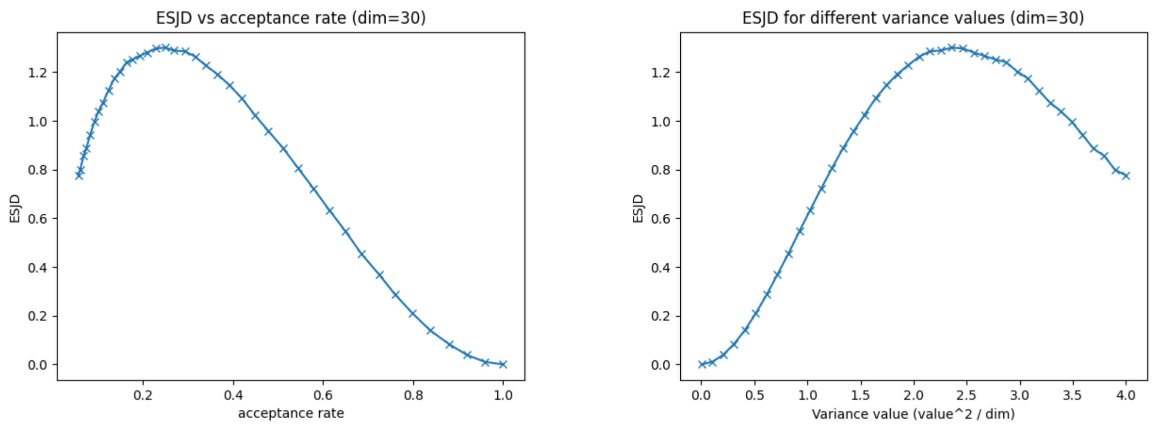


Figure 27: ESJD against acceptance rate and variance. $(d = 30)$

but that jumping between modes still happens frequently enough. For $d = 20$, we set $c = 7.5$. For $d = 30$, we set $c = 5.0$. Then we set our covariance matrices in Equation 3 to be the identity matrix, so $\Sigma_1 = \Sigma_2 = \Sigma_3 = I_d$. Overall, this experiment had 200 simulations which each ran for 100,000 iterations.

This experimentation in $d = 20, 30$ yields a very exciting result. On adjusting c close enough, ESJD continues to be maximised at an acceptance rate of around 0.234 and the optimal variance is approximately $2.38^2/d$. This is a meaningful result because we find that the Markov chain is exploring all three modes through the histogram and traceplot in Figure 28. Clearly, even though the i.i.d. assumption does not hold, the 0.234 figure still seems to be quite useful here.

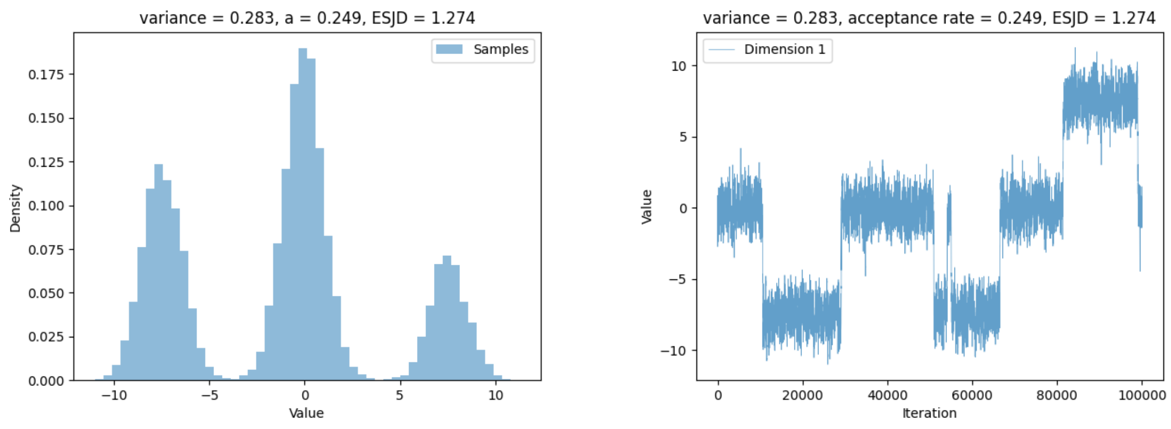


Figure 28: Histogram and traceplot of the first component.

5.2.2 Different diagonal covariance matrices with scaling factors in each component

What if we use different scaling factors in the covariance matrices Σ_i? In most real-world data distributions, the variance of each variable is not likely to be the same. We repeat this experiment with with different covariance matrices for each mode, where each mode has the form $diag(c_1, c_2, \ldots, c_d)$ where c_i are i.i.d. samples from the Uniform$[0, 2]$ distribution such that $\mathbb{E}[c_i] = 1$ and $\mathbb{E}[c_i^2] < \infty$.

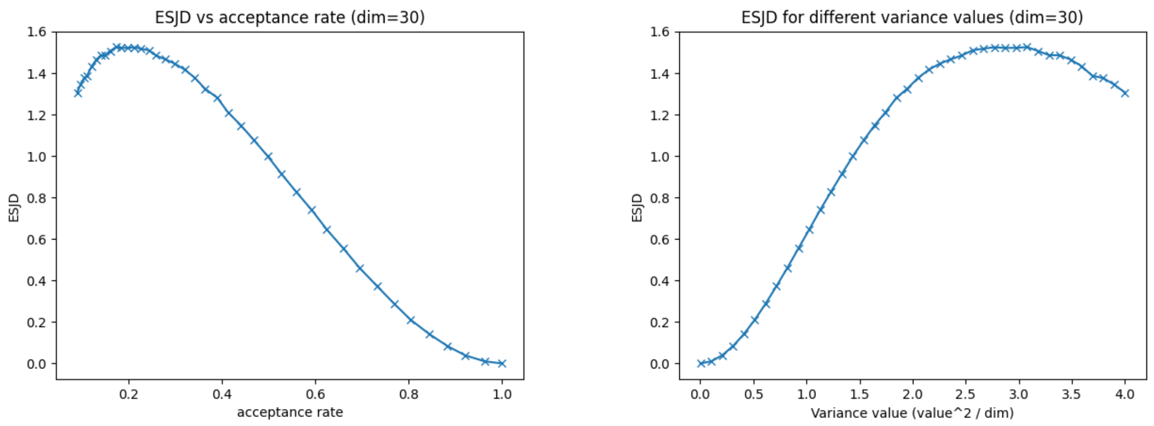


Figure 29: ESJD against acceptance rate and variance for scaled covariance matrices. $(d = 30)$

Here, we find the theoretical figures match the results somewhat, but there is still a clear discrepancy. As shown in Figure 29, the optimal acceptance rate here is 0.205 and the optimal variance is $2.462^2/d$. Most notably, the downward trend of ESJD has started well before reaching the 0.234 acceptance rate.

5.3 Discussion

In this section, we conducted experiments on a variety of multimodal distributions and found that the optimal acceptance rate that maximises ESJD is still approximately 0.234 even in cases like Section 5.2 where the target density is not a product of i.i.d. single-dimensional component densities. However, the optimal variance scaling factor being $2.38^2/d$ is not necessarily true in dimensions as low as $d = 20, 30$ even in cases where the theory should apply, such as Section 5.1. Furthermore, the 0.234 acceptance rate was no longer optimal for the case of i.i.d. sampled scaling factors in each dimension, with a significant deviation to 0.205. Future work might entail investigating whether these deviations from expected theoretical results are a practical problem with multimodal distributions, or if this is simply due to the lower dimension of these experiments.

The struggles we encountered with multimodal distributions naturally segue into the next chapter of experiments and discussion on the Parallel Tempering method, which is designed to improve exploration in multimodal target distributions. That being said, it is also worth investigating alternative proposal distributions to the Gaussian, such as an adaptive hollow ball or ring centred at the current point $\mathbf{x}$. It might also be interesting to investigate different Metropolis algorithms, such as the Repelling-Attracting Metropolis algorithm.

6 Parallel Tempering

6.1 Introduction

Parallel tempering, also known as replica exchange Monte Carlo, is a technique used to improve the convergence of MCMC methods, particularly in cases where the target distribution has multiple modes. In traditional MCMC methods like the Metropolis algorithm, a single Markov chain is used to explore the state space and sample from the target distribution. However, we might get stuck in local modes of the target density, especially when modes are far away. Parallel tempering uses multiple copies of the MCMC chain, each of which explores the target distribution at a different temperature T. Here, temperature is a parameter that controls the shape of the energy landscape of the target distribution. The inverse temperature, typically denoted by β, is the reciprocal of the temperature, i.e. $\beta = \frac{1}{T}$.

Higher temperatures (smaller β) correspond to smoother, more uniform landscapes, which allows the chain to explore a larger portion of the state space more freely and hop between different modes of the distribution. Lower temperatures (larger β) correspond to rougher, more peaked landscapes yielding more similar behaviour to the original Metropolis algorithm, where a chain explores the details of the target distribution within a specific mode.

The key idea behind parallel tempering is to allow copies to occasionally exchange states with each other based on some probability in order to escape local modes and mix faster.

6.1.1 Problem Description and Algorithm

We want to sample from a target distribution π which has a target density $f_d(\mathbf{x})$ on state space χ. That is to say, $\mathbf{x} \in \mathbb{R}^d$.

We have a sequence of tempered target distributions $(f_d(\mathbf{x}))^{\beta_j}$ where $0 \leq \beta_n < \beta_{n-1} < \cdots < \beta_1 < \beta_0 = 1$ are the *inverse temperature values*. We require that $(f_d(\mathbf{x}))^{\beta_0} = f_d(\mathbf{x})$ (that is, that $\beta_0 = 1$). For simplicity, we assume the tempered target distributions are powers of the original density for all parallel tempering experiments.

Then, we construct a new (unnormalised) stationary density $\prod_{j=0}^{n}(f_d(\mathbf{x}_j))^{\beta_j}$. We have one chain at each of the $n+1$ values of β, and we take each $\mathbf{x}_j \in \mathbb{R}^d$ corresponding to each chain at that fixed inverse temperature β_j. After a long time, the $\beta_0 = 1$ chain, also called the "cold" chain, with its stationary density $(f_d(\mathbf{x}_0))^{\beta_0}$ should correspond to the original target density of interest $f_d(\mathbf{x})$, and the idea is that other chains with higher temperatures (lower β) will speed up the convergence of the values $\mathbf{x}_0$ to the original target density $f_d(\mathbf{x})$.

In each iteration of the algorithm, we alternate between two types of moves:

- Within temperature move: take a typical random walk update within each $(f_d)^{\beta_j}$.
- Swap: for two chains with β_j, β_k inverse temperature values, switch the chain values of $\mathbf{x}_j$ and $\mathbf{x}_k$ with probability

$$\min(1, \frac{(f_d(\mathbf{x}_k))^{\beta_j}(f_d(\mathbf{x}_j))^{\beta_k}}{(f_d(\mathbf{x}_j))^{\beta_j}(f_d(\mathbf{x}_k))^{\beta_k}}) \tag{4}$$

6.1.2 Optimal Scaling for Parallel Tempering

Since the hot chains can explore the state space more quickly and mix better, we would like to maximise how much the hot chain influences the cold chain; i.e. how frequently we can swap values from the hottest chain to the coldest chain so that the cold chain can mix faster and escape local modes. To do this, we want to maximise the effective speed with which the chain values move along in the inverse temperature domain. The spacings of the inverse temperatures β are crucial to this efficiency. If β_j and β_k are too far apart, we usually reject these swaps described by Equation 4, but if they are too near, the swaps will not improve mixing. Therefore, we would like to have it somewhere in the middle. A surprising and interesting result is that spacing the inverse temperatures such that the swap acceptance probability is approximately 0.234 is also optimal [2] under certain conditions.

6.1.3 Expected Squared Jumping Distance

As described in the previous section, we want to space out our inverse temperatures such that we swap just the right amount. The expected squared jumping distance for parallel tempering thus refers to the expected squared jump in inverse temperatures. Formalising this, when we attempt to swap the chain values between the inverse temperatures β and $\gamma := \beta + \epsilon$ where $\beta, \epsilon > 0$ and $\beta, \gamma \leq 1$, the swap is either accepted, in which case the values move a squared distance of

$(\gamma - \beta)^2 = \epsilon$, or the swap is rejected, in which case the distance moved is 0. This leads to a very natural definition for ESJD, which is

$$ESJD = \mathbb{E}[(\gamma - \beta)^2] = \epsilon^2 \times \mathbb{E}[\Pr(\text{swap accepted})]. \tag{5}$$

Maximising ESJD effectively maximises the efficiency of the attempted swap moves in providing mixing in the inverse temperature space, or in other words, maximises the speed with which the chain values move in the inverse temperature space.

6.2 Methodology

For the parallel tempering experiments, we wanted to investigate how well the 0.234 optimal swap acceptance rate theory would hold for the various multimodal target distributions in Section 5. After all, the parallel tempering method is highly effective for exploring multimodal distributions [7][2].

We ran experiments that examined the trend of ESJD with swap acceptance rate for each of the "rough carpet" distribution from Section 5.1, three-mixture distribution from Section 5.2, and standard multivariate Gaussian distribution as a baseline for comparison. For each distribution, we ran many simulations with dimension $d = 20$ where each simulation was a parallel tempering algorithm with 40 different average swap acceptance rate values from 0.01 to 0.8. We explain how we set up these algorithms to match these swap rates in Section 6.2.1. Each parallel tempering algorithm ran for 20,000 iterations, so every individual chain in the algorithm took 20,000 steps. We attempt a swap every 20 steps. Furthermore, we ran each algorithm instance over 3 different seeds and average the results to reduce the effects of randomness caused by a particular seed. Overall, each experiment on a single distribution had 120 simulations which each ran for 20,000 iterations per chain, and the number of chains for each parallel tempering algorithm simulation was determined by the inverse temperature spacing given by the intended average swap acceptance rate. We refer the reader to Section 6.2.1 for details on how these inverse temperatures were set up.

For each target distribution, we set the tempered target distribution to simply be the original distribution raised to the power of β as described in Section 6.1.1. set the modes to be far away enough such that the other modes were unreachable by the standard random-walk Metropolis algorithm after 100,000 iterations, but still reasonably close so that the parallel tempering method would show the cold chain values would swap between modes somewhat frequently.

6.2.1 Constructing an inverse temperature ladder

There are two ways to construct an inverse temperature ladder. The first way is to simply select the inverse temperatures using a geometric series spacing. However, since we are examining the 0.234 swap acceptance rule, we need to have a way of constructing the inverse temperature spacings such that the probability of a swap between adjacent chains is approximately 0.234 (or any other value). We do this iteratively using a slightly modified version of the procedure in Atchadé, Roberts, & Rosenthal (2011) Section 2.2 as follows.

We start the construction with an initial inverse temperature $\beta_0 = 1$ and a minimum value $\beta^* = 0.01$. We construct the β's iteratively starting from $\beta_1, \beta_2, \ldots$ We denote the current most recent β that is already added to the inverse temperature ladder as β_{curr} and we would like to add a new β^* to the ladder. Suppose we would like to space our β's by a swap acceptance rate of $s = 0.234$, but you could set s how you like. We initialize $\rho_n = 0.5$ where $n = 1$ initially is the number of iterations of an indefinitely running while loop, and set $\beta^* = \beta_{curr}(1 + e^{\rho_n})^{-1}$.

To determine whether this value of β^* should be added to the ladder, we enter the while loop and draw 1000 samples from the target distribution tempered by β^*, and 1000 samples from the target distribution tempered by β_{curr}. We draw these samples via heuristics that match the target distribution as the number of samples goes to ∞. Then, for each of the 1000 samples drawn, we calculate the swap probability of the samples from the respective β^* and β_{curr} chains, and average this out. If the average swap probability a is within a certain margin ($s \pm 0.005$) of the desired spacing of the β's, then we add β^* to the ladder and set $\beta_{curr} = \beta^*$. Otherwise, we set a new value for $\rho_n = \rho_{n-1} + n^{-0.25}(a - s)$, and calculate the new value of $\beta^* = \beta_{curr}(1 + e^{\rho_n})^{-1}$. This recurrence ensures that if $a > s$ then β^* decreases to make the distance between β's further apart, and vice versa. We then repeat the loop to determine whether the new value of β^* should be added to the ladder.

We continue adding β's until we reach some minimal inverse temperature where mixing is deemed sufficiently fast, at which point we take the minimal inverse temperature as our final β value. For our experiments, we set the minimal value to be 0.01.

6.3 Multivariate Gaussian Distribution

We first give an example of the standard multivariate Gaussian distribution to verify the correctness of our implementation. Note that the standard multivariate Gaussian as a target distribution satisfies the necessary assumptions of Theorem 1 in Atchadé, Roberts, & Rosenthal (2011): our target is a product of individual component densities, and we temper the distributions by raising

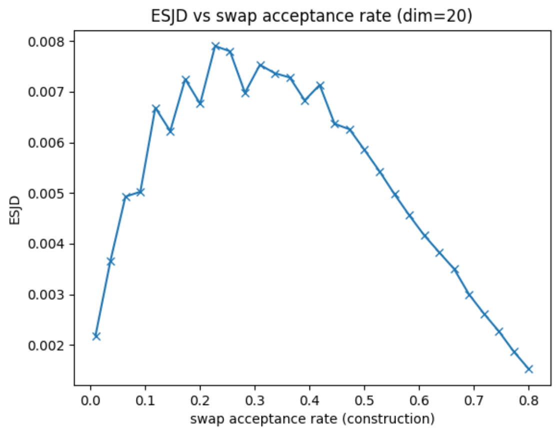


Figure 30: ESJD vs swap acceptance rate for parallel tempering multivariate Gaussian target.

them to the power of the original density, and $\epsilon \searrow 0$ such that $\epsilon = d^{-1/2}\ell$ for some positive constant ℓ. We find that the results are consistent with the theory, and that the optimal swap acceptance rate is around 0.234. Please refer to Figure 30.

6.4 "Rough Carpet" Distributions

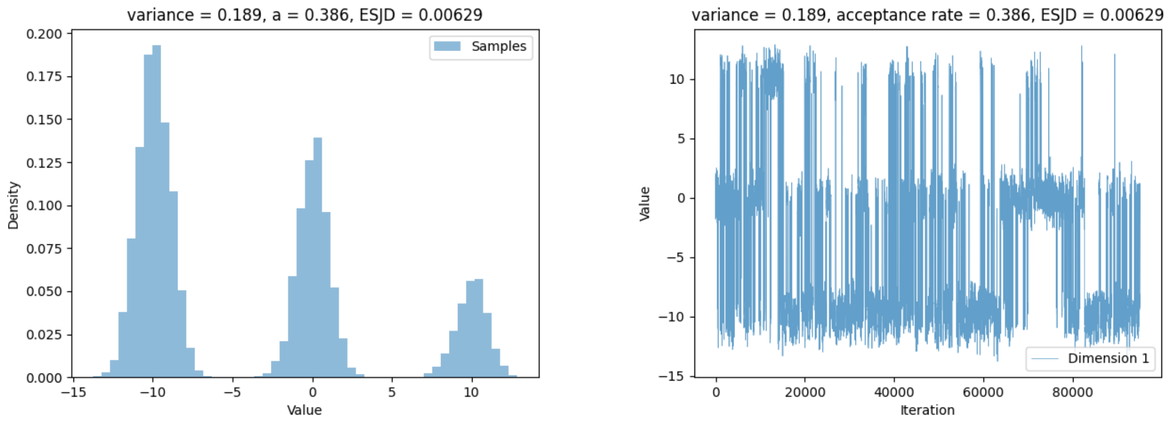


Figure 31: Parallel tempering histogram and traceplot of the cold chain first component in the carpet distribution.

In this section, we use the same target distribution as described in Section 5.1. Here, we examine $f_d(\mathbf{x}) = \prod_{i=1}^{n} f(x_i)$. Our individual component density $f(x)$ is the same as Equation 2, and in the $f(x)$ definition we set $m_1 = -10$, $m_2 = 0$, $m_3 = 10$. We provide an example histogram and traceplot after 100,000 iterations with a constructed ladder to demonstrate what mixing looks like. Unlike the standard random-walk Metropolis which cannot escape from the central mode in this setting of the target distribution, the cold chain swaps between modes considerably often. Refer to Figure 31 to see the cold chain's mixing in the first component (all components are i.i.d.).

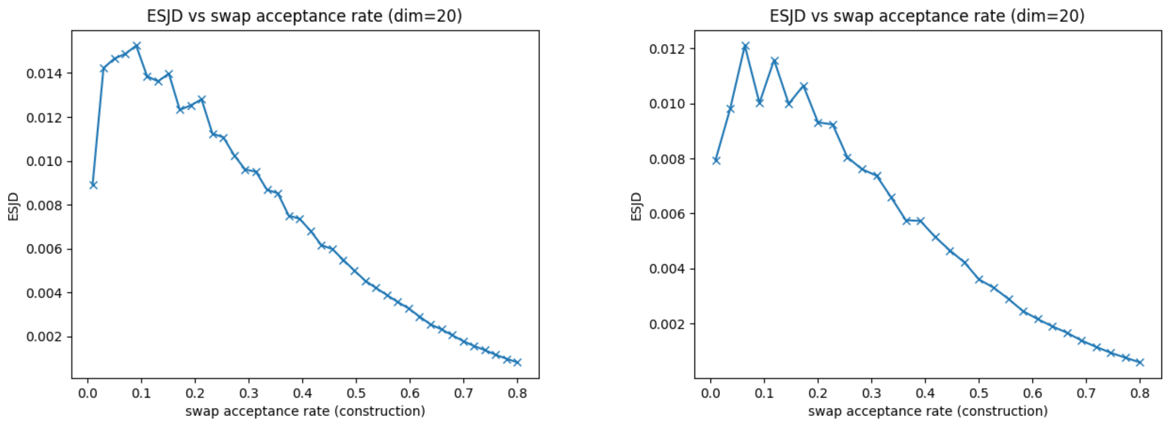


Figure 32: Two separate experiments of ESJD vs swap acceptance rate for parallel tempering rough carpet target.

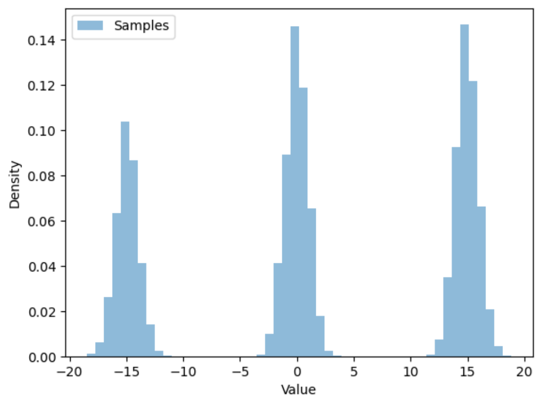
Like the multivariate Gaussian, this rough carpet target distribution satisfies the requirements of the theorem. Unlike the multivariate Gaussian, the results are quite different. We report the results of two separate instances of this experiment on the rough carpet distribution in Figure 32. The optimal swap acceptance rate that maximises the ESJD is far from 0.234. Here, in both experiments, the optimal acceptance rate corresponds to about 0.1. This might be due to the awkward shape of the distribution. Since it has 3^d modes, the density at each mode cannot be that high, which might mean the target distribution of a hot chain would look almost uniform and thus hot chains might not be sampling the right values that need to be swapped to colder chains.

6.5 Three Mixture Distributions

In this section, we use the same target distribution as described in Section 5.2 Equation 3. We set $m_1 = (-15, 0, 0, \ldots, 0)$, $m_2 = (0, 0, \ldots, 0)$, $m_3 = (15, 0, 0, \ldots, 0)$ and $\Sigma_1 = \Sigma_2 = \Sigma_3 = I_d$.

We provide an example histogram and traceplot after 100,000 iterations with a constructed ladder to demonstrate what the cold chain mixing looks like in the first component in Figure 33. Although this has not yet converged to the target distribution, the cold chain swaps between modes considerably often, unlike the standard random-walk Metropolis which cannot escape from the central mode with this setting of the target distribution.

We find that the swap acceptance rate value near 0.234 is not actually the one that maximises



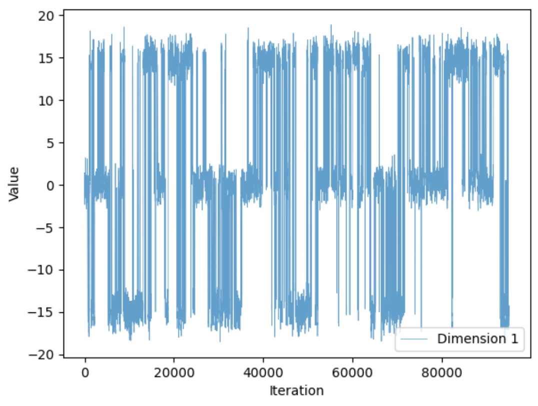


Figure 33: Parallel tempering histogram and traceplot of the cold chain first component in the Three-Mixture distribution.

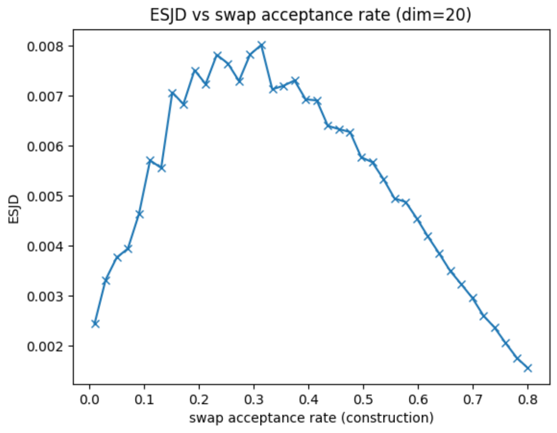


Figure 34: ESJD vs swap acceptance rate for parallel tempering three-mixture target.

ESJD; rather, it is the third highest in Figure 34. The highest value of ESJD corresponds to a swap acceptance rate of approximately 0.314, and the second-highest corresponds to 0.294. This result might be because the target distribution is not a product form, which is one of the key assumptions required for the optimal spacing result for parallel tempering.

6.6 Discussion

Parallel tempering proves to be a much more useful method than the standard random-walk Metropolis algorithm. Having tested this extensively, parallel tempering is extremely good at exploring modes that are practically impossible for a random-walk algorithm to reach in a reasonable amount of time.

However, the optimal swap acceptance rate figure for parallel tempering is not as flexible as the optimal acceptance rate for random walk Metropolis. The 0.234 figure is not obviously relevant to the three-mixture distribution which does not follow the product form, or even to the rough carpet distribution which does have this i.i.d. product form.

The most immediate area where this could be improved is by running these experiments for a larger number of iterations, both in the sampling from the target density for the construction of the temperature ladders and the actual number of iterations for each parallel tempering algorithm.

Firstly, in constructing the temperature ladders, we only drew 1000 samples, which may be insufficient to provide a good representation for certain distributions such as the rough carpet distribution, which has 3^d modes. It is not very sensible to draw 3^{20} samples, but there might yet be a better way to construct the inverse temperature spacings than the method we have described. It might make sense to construct many inverse temperature ladders for the same desired spacing and take the average of them, but constructing a single ladder as described in Section 6.2.1 is already quite costly.

Next, while 20,000 iterations of an algorithm is good enough for a lot of mixing to occur, it typically takes many more iterations for the samples to converge to some complicated target distributions. Therefore, the experiment could be repeated with a higher number of runs per algorithm, and have more seeds than just 3. However, given the limits of our personal hardware, the current experiment is already extremely expensive, and takes around 8-12 hours on a M2 Macbook Air while running as a high-priority process.

Other promising avenues for future work include investigating whether the 0.234 figure generalises better outside the assumptions in higher dimensions than 20 or 30, finding a computationally cheaper and/or a more deterministic method to construct the inverse temperature ladder, and parallelizing the chains to speed up the run-time of the experiment.

7 Conclusion

In summary, our investigation into optimal scaling for the Metropolis algorithm reveals consistent findings across various scenarios. For Gaussian target distributions with i.i.d. components, an acceptance rate of approximately 0.234 optimizes ESJD, regardless of the starting point or dimensionality. Furthermore, a proposal variance of $2.38^2/d$ generally maximises efficiency. While the optimal 0.234 acceptance rate still maximise ESJD for Gaussian distributions with non-i.i.d. components, this acceptance rate and a maximised ESJD do not necessarily result in effective sampling.

We also investigate different proposal distributions using the standard random walk Metropolis algorithm and find that multiple proposal distributions still maximise the ESJD at an acceptance rate of 0.234 as long as we tune the proposal variance appropriately.

In multimodal distributions, the 0.234 acceptance rate appears optimal for standard RWM even in some general cases where the target density is not a product of i.i.d. single-dimensional component densities, but the 0.234 acceptance rate does not appear optimal for the case of a multimodal distribution with many modes where we have i.i.d. sampled scaling factors in each dimension, with a significant deviation to 0.205. Furthermore, the optimal variance scaling factor being $2.38^2/d$ is not necessarily true in dimensions as low as $d = 20, 30$ even in some cases where the theory should apply. These challenges in exploring multimodal distributions lead us to the Parallel Tempering method, which proves superior in this task. However, the optimal swap acceptance rate of 0.234 in Parallel Tempering is not as widely applicable outside the theorem assumptions compared to the standard random walk Metropolis.

References

[1] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1):5–43, January 2003.

[2] Yves F. Atchadé, Gareth O. Roberts, and Jeffrey S. Rosenthal. Towards optimal scaling of metropolis-coupled Markov chain Monte Carlo. *Statistics and Computing*, 21(4):555–568, October 2011.

[3] Mylène Bédard. Efficient sampling using metropolis algorithms: Applications of optimal scaling results. *Journal of Computational and Graphical Statistics*, 17(2):312–332, 2008.

[4] Andrew Gelman, Walter R Gilks, and Gareth O Roberts. Weak convergence and optimal scaling of random walk metropolis algorithms. *The annals of applied probability*, 7(1):110–120, 1997.

[5] Andrew Gelman and Cristian Pasarica. Adaptively Scaling the Metropolis Algorithm Using Expected Squared Jumped Distance. *SSRN Electronic Journal*, 2007.

[6] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli*, pages 223–242, 2001.

[7] W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970. Publisher: [Oxford University Press, Biometrika Trust].

[8] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling for various metropolis-hastings algorithms. *Statistical science*, 16(4):351–367, 2001.

[9] Jeffrey Rosenthal. *Optimizing and Adapting the Metropolis Algorithm*. 03 2014.

[10] Jun Yang, Gareth O. Roberts, and Jeffrey S. Rosenthal. Optimal scaling of random-walk metropolis algorithms on general target distributions, 2020.