Sampling by Divergence Minimization

Ameer Dharamshi*†[1], Vivian Ngo[1], and Jeffrey S. Rosenthal[1]

[1]Department of Statistical Sciences, University of Toronto,
Toronto, ON, Canada

May 2, 2021

Abstract

We introduce a family of Markov Chain Monte Carlo (MCMC) methods designed to sample from target distributions with irregular geometry using an adaptive scheme. In cases where targets exhibit non-Gaussian behaviour, we propose that adaption should be regional in nature as opposed to global. Our algorithms minimize the information projection side of the Kullback-Leibler (KL) divergence between the proposal distribution class and the target to encourage proposals distributed similarly to the regional geometry of the target. Unlike traditional adaptive MCMC, this procedure rapidly adapts to the geometry of the current position as it explores the space without the need for a large batch of samples. We extend this approach to multimodal targets by introducing a heavily tempered chain to enable faster mixing between regions of interest. The divergence minimization algorithms are tested on target distributions with multiple irregularly shaped modes and we provide results demonstrating the effectiveness of our methods.

Keywords: adaptive MCMC, multimodal targets, KL divergence, sampling, tempering

Declarations:

Funding: No funding was received to assist with the preparation of this manuscript.

Conflicts of Interest/Competing Interests: The authors have no conflicts of interest to declare that are relevant to the content of this article.

Availability of data and material: Not applicable.

Code availability: The implementations of the algorithms discussed here along with all code used to generate examples can be accessed at `https://github.com/AmeerD/Scout-MCMC`

*Corresponding Author (e-mail: ameer.dharamshi@mail.utoronto.ca)

†ORCID iD: 0000-0002-5505-4765

Contents

1 Introduction **3**
1.1 Related Works 4

2 Divergence Minimization Sampler **6**
2.1 Objective 6
2.2 Algorithm Details 9
2.3 Divergence Minimization: A Case Study 9
2.4 Convergence and Finite Adaptation 12

3 Scout MCMC **15**
3.1 Motivation 16
3.2 Algorithm Details 16
3.3 Finite Adaptation 17

4 Examples **19**
4.1 Double Banana Distribution 20
4.2 Basis Vector Distribution 21
4.3 Banana Bunch Distribution 24

5 Discussion and Future Work **27**

A Appendix **29**
A.1 Approximating the Gradient 29

1 Introduction

Markov Chain Monte Carlo (MCMC) is a class of algorithms designed to efficiently and effectively sample from a diverse set of target distributions [Brooks et al., 2011]. Classical MCMC methods perform excellently when the target is well-behaved and unimodal. However, when targets exhibit unusual geometry or have multiple modes, core techniques such as random walk Metropolis (RWM) tend to perform poorly. These are the challenges that motivate much active MCMC research. In this paper, we propose a set of algorithms that specifically aim to effectively sample from unusual target distributions, with extensions to the multimodal realm.

For targets with atypical geometry, adaptive MCMC as a class of methods has proven to outperform classical MCMC [Atchadé et al., 2011, Haario et al., 2001, Andrieu and Thoms, 2008]. One of the core ideas driving adaptive MCMC is that a proposal distribution that is similar in shape to the target function will produce higher quality samples than a generic proposal. In adaptive Random Walk Metropolis (aRWM), this customization is accomplished by proposing with the empirical covariance matrix of the samples produced up to the current iteration. As the algorithm progresses, the proposals improve. Eventually, the empirical covariance matrix approaches the hypothetical global optimal sampler. Convergence to the target distribution can be upheld using the principles of containment and diminishing adaptation, or finite adaption [Roberts and Rosenthal, 2007, Rosenthal, 2011].

However, aRWM does have its limitations. When the target distribution exhibits highly irregular, non-Gaussian geometry, a single optimal Gaussian proposal distribution as used by aRWM may not perform well in all regions of the target distribution.

With multimodal targets, the challenges of sampling from unimodal targets are compounded by low probability regions that may exist between modes. Parallel tempering is a commonly used, generally applicable method for this class of targets. In parallel tempering, multiple chains are run simultaneously on the target with different levels of tempering. Intuitively, it is easier to cross low probability barriers when a high tempering factor flattens the target. The positions of chains are randomly swapped to allow the non-tempered chain to move between modes. This realtively simple procedure displays remarkable mode discovering capabilities [Swendsen and Wang, 1986, Geyer, 1991].

Similar to aRWM, parallel tempering may not perform optimally when the target's modes have distinct, irregular geometry. In addition to the challenge of sampling from irregular, non-Gaussian geometry, parallel tempering also risks swapping out of a given mode before fully exploring it.

In this work, we expand on the idea that a global optimal proposal distribution may not be sufficient and we instead discuss the idea of region-specific sampling. We introduce a set of related algorithms designed to sample effectively from unusual geometries by exploiting local information about the target distribution. Instead of waiting for samples to be produced in order to trigger adaptation, we use ideas from the recently popular stochastic variational

inference class of methods [Salimans et al., 2015]. By measuring the similarity between the target and proposal distributions using the Kullback-Leibler (KL) Divergence, we use gradients to devise an update rule that is not reliant on having a large batch of samples to work with.

More specifically, to construct proposals that are shaped similarly to the target, we minimize the information projection, or I-projection, side of the KL Divergence [Yamano, 2009]. First, let's define the target distribution as p, and the family of proposal distributions to be $q \in \mathcal{Q}$. The I-projection, $\mathcal{D}(q||p) = E_q\left[\log \frac{q}{p}\right]$, tends to produce an underdispersed q that locks onto a specific mode of p [Murphy, 2012]. In other words, through minimization, it produces a distribution similar to the local geometry of p. Such a distribution is equipped to rapidly produce samples from oddly shaped regions of a target distribution. We term this approach the Divergence Minimization (DM) Sampler.

In order to use the DM sampler in a multimodal setting, we propose an extension, Scout MCMC, which is designed to sample from multimodal distributions. Scout MCMC relies on the DM sampler and a single highly tempered "scout" chain to explore the global space and search for modes while the DM sampler generates local samples from the non-tempered main chain. Occasionally, the positions of the two chains can be swapped, allowing the DM sampler to access new regions. In contrast with parallel tempering, Scout MCMC only requires two chains in total, whereas parallel tempering tends to require many more chains to be effective.

Finally, we recognize that at each iteration, the covariance matrix produced by the gradient update rule represents a proposal distribution adept at sampling from its local region. This generated proposal distribution can reasonably be used for nearby points, assuming some degree of continuity. Thus, we introduce a two-stage extension to the DM sampler and Scout MCMC. In the first stage, we gather proposal distributions using the DM sampler or Scout MCMC. Next, we use these proposal distributions to characterize a non-adaptive Metropolis-Hastings algorithm in the second phase.

Before discussing the specifics of the algorithms in Section 2 and Section 3, we first discuss a number of relevant related works.

1.1 Related Works

The DM sampler draws inspiration from Titsias and Dellaportas [2019]. In this paper, the authors optimize an objective function composed of the product of the entropy function and the average proposal acceptance rate. The proposals for the adaptive MCMC algorithm are then based off of gradient updates that seek to maximize this function, producing a wide range of proposals via the entropy function, while maintaining a reasonable acceptance rate. This algorithm is able to outperform other MCMC algorithms, including Hamiltonian Monte Carlo schemes. In addition to supporting significant adaptation at early stages of a chain, Gradient-based Adaptive MCMC also allows for adaptation upon rejecting a proposal, a noteworthy feature as most adaptive algorithms do not

directly consider the information offered by rejected samples.

While the entropy function is a general function applied to the entire distribution, the algorithm presented in this paper is based on the premise that in cases with difficult geometry, it is necessary to focus on specific local regions while sampling instead of the entire target distribution at once. This is accomplished by leveraging the I-Projection of the target over the set of proposal distributions. The I-projection underestimates the support of the target distribution and will hone in on one mode as opposed to the entropy function which attempts to discover a range of samples from the entire target function at once [Shannon, 1948, Murphy, 2012]. By using the I-projection, the algorithm can produce quality samples from specific modes without spending too much time in that mode. This regional behaviour helps to overcome limitations in the Gaussian function class typically used for proposal distributions by reducing the current region of interest into manageable pieces.

Parallel tempering is another candidate method, and is related to our Scout MCMC algorithm. In parallel tempering, multiple chains are run simultaneously on the target distribution with different levels of tempering applied. The intuition behind parallel tempering is that in the highly tempered chains, it will be easier to cross low-probability boundaries which can subsequently be randomly swapped with the non-tempered chain for between-mode mixing [Swendsen and Wang, 1986, Geyer, 1991]. Similarly, the moderately tempered chains improve mixing within regions.

However, parallel tempering does have its limitations. From a computational perspective, executing many chains but ultimately only using the samples from the non-tempered chain is burdensome. Secondly, while parallel tempering is excellent at moving between modes, it does not address the risk of potentially leaving a mode before fully exploring it. Scout MCMC seeks to address these challenges. Noting that the higher the tempering, the easier it is for a chain to cross low probability boundaries, Scout MCMC uses a single highly tempered chain to move between modes instead of many tempered chains, thus reducing the computational burden. Then, we exchange the non-tempered chain with the DM sampler to improve local sampling so that regions can be better explored between swap moves. Given the use of a highly tempered chain in Scout MCMC, one could thus consider Scout MCMC as a special case of parallel tempering.

Using parallel chains for similar purposes, in Craiu et al. [2009], the authors introduce the algorithm Inter-chain Adaptation (INCA), which uses multiple stages of sampling. The first stage involves sampling the state space with parallel chains to determine the locations of modes and to partition the state space, while the second stage uses these predetermined modes as a guide to sample from the target distribution. The acceptance probabilities of new proposed points then depend on the region in which the current and proposed points reside. In order for the algorithm to be robust for distributions with many modes, it is essential to include the additional chains in the first phase to explore the entire space.

Finally, the Jumping Adaptive Multimodal Sampler (JAMS) algorithm addresses the challenges of multimodal sampling by front loading the computational burden of mode discovery by using optimization techniques to search for

modes, and subsequently incorporates this information into the sampling phase [Pompe et al., 2019]. In the sampling phase, dedicated "jump moves" are used to move between modes directly. Once in a mode, any sampler adept at unimodal sampling can be employed.

Both INCA and JAMS rely on the results of a front loaded mode discovery phase for between mode mixing. In the event these first stages are unable to completely explore the space and identify every mode, the algorithms may produce biased samples. In an effort to prioritize flexibility and limit assumptions, the boundaries between modes in our algorithms are not defined and our algorithm is designed to find modes during sampling with the heavily tempered scout chain. This decision allows for mode discovery at any point in the algorithm.

2 Divergence Minimization Sampler

We propose that the challenge of sampling from irregular geometries can be overcome by focusing on smaller regions of a given target distribution. These more manageable pieces can be characterized by simpler features that can be adequately sampled from using common proposal distributions such as the Gaussian proposal. This region specific sampling scheme requires addressing two core issues: identifying regions of interest and determining how best to sample from these regions. At the most granular level, each individual point in the space could constitute its own region. The rationale is that every point has its own unique surrounding geometry and thus there exists some optimal way to generate a new sample when starting at each and every point.

The latter challenge characterizes the problem of identifying this optimal sampling procedure. To tackle this issue at the granular level, we propose using the I-projection component of the KL Divergence as a similarity measure between the target and proposal distributions to construct proposals with similar geometry to the region around the current point [Murphy, 2012]. Defining the target distribution as p and the family of proposal distributions to be $q \in \mathcal{Q}$, the I-projection is $\mathcal{D}(q||p) = E_q\left[\log \frac{q}{p}\right]$. In the context of an MCMC proposal, we consider the family of proposal distributions to be Gaussian and our objective is to determine the covariance matrix that characterizes the Gaussian with minimal divergence from the target distribution at the current point. Such a proposal can be defined as:

$$q(y|x) \sim N\left(x, LL^T\right)$$

where x is the current position, $y = x + L\epsilon$ is the proposal, $\epsilon \sim N(0, 1)$ and L is the Cholesky factor of the proposal covariance matrix [Higham, 2009].

2.1 Objective

To find a proposal distribution that minimizes the divergence with the local geometry of the target distribution, we consider using gradient updates performed

at each iteration of the MCMC chain. At the same time, we must be cognizant of the acceptance rate. In essence, we want to have both a small I-projection so that the proposal and the target are similar, as well as a reasonably high acceptance rate so that we are able to use the samples from our proposals. As such, we propose the following as an objective function that balances both the exponential of the negative I-projection and the average acceptance rate of the proposal:

$$s(x) = \exp\left[-\beta \mathcal{D}(q||p)\right] \cdot \int \alpha(x, y; L) q(y|x) dy$$

In the above, β is a hyperparameter that balances the impact of the I-projection with the average Metropolis acceptance rate defined by:

$$\alpha(x, y; L) = \min\left\{1, \frac{p(y)}{p(x)}\right\}$$

where x is the current position, y is the proposal, and L is the proposal distribution Cholesky factor [Brooks et al., 2011]. Notice the negative inside the exponential term of $s(x)$. As the I-projection is non-negative, the negative exponent bounds the exponential term between 0 and 1 with the maximum obtained when $\mathcal{D}(q||p) = 0$. Also note that the average acceptance rate ranges between 0 and 1. As a result of these bounds, $s \in [0, 1]$ and is maximized when we have high acceptance rates with a proposal that is similar to the target. Thus, the problem of identifying a suitable proposal distribution has been reduced to maximizing $s(x)$ where the optimal proposal distribution at any given x can be characterized by the corresponding optimal Cholesky factor L_x at the global optimum.

To make the objective function easier to manipulate, instead of optimizing $s(x)$, we can optimize the logarithm of $s(x)$. That is:

$$\begin{aligned}
\log s(x) &= -\beta D(q||p) + \log \int \alpha(x, y; L) q(y|x) dy \\
&= -\beta E_q \left[\log \frac{q(y|x)}{p(y)}\right] + \log E_q \left[\alpha(x, y; L)\right] \\
&= \beta E_q \left[-\log q(y|x)\right] + \beta E_q \left[\log p(y)\right] + \log E_q \left[\alpha(x, y; L)\right] \\
&= \beta H_q + \beta E_q \left[\log p(y)\right] + \log E_q \left[\alpha(x, y; L)\right]
\end{aligned}$$

The above statement of $\log s(x)$ contains expectations entangled with both the p and q distributions that precludes a closed form solution. In particular, notice that the final term is the logarithm of an expectation. Such a term is certainly not ideal for optimization purposes. The most advisable path forward to maximize $\log s(x)$ is to instead bound it below using Jensen's inequality. We can then optimize the lower bound instead of the objective directly. Thus we have:

$$
\begin{aligned}
\log s(x) &\geq \beta H_q + \beta E_q \left[\log p(y)\right] + E_q \left[\log \alpha(x, y; L)\right] \\
&= \beta H_q + \beta E_q \left[\log p(y)\right] + E_q \left[\log \min \left\{1, \frac{p(y)}{p(x)}\right\}\right] \\
&= \beta H_q + \beta E_q \left[\log p(y)\right] + E_q \left[\min \left\{0, \log p(y) - \log p(x)\right\}\right] \\
&= \beta H_q + \beta E_\epsilon \left[\log p(x + L\epsilon)\right] + E_\epsilon \left[\min \left\{0, \log p(x + L\epsilon) - \log p(x)\right\}\right] \\
&=: \mathcal{J}(x)
\end{aligned}
$$

We can now use $\mathcal{J}(x)$ as a lower bound for the objective function and optimize it. However, while $\mathcal{J}(x)$ is certainly simpler than $\log s(x)$, a general closed form solution of the maximum at each value of x is still not attainable. Instead, we turn to iterative optimization methods. We choose gradient ascent as a generally accessible method to maximize $\mathcal{J}(x)$.

Gradient ascent requires specifying the gradient of $\mathcal{J}(x)$ with respect to the Cholesky factor L. We leave the detailed derivation of the approximate gradient of $\mathcal{J}(x)$ to Appendix A.1 and present the final result here:

$$
\begin{aligned}
\nabla_L \mathcal{J}(x) = \beta \text{diag}\left(\frac{1}{L_{11}}, \ldots, \frac{1}{L_{kk}}\right) + \sum_{j=1}^{J} \frac{\beta}{p(x + L\epsilon_j)} p'(x + L\epsilon_j)\epsilon_j^T + \\
\sum_{j=1}^{J} \nabla_L \min \left\{0, \log p(x + L\epsilon_j) - \log p(x)\right\}.
\end{aligned}
$$

where x is the current position, L is the current value of the Cholesky factor, and ϵ_j are a sample of J standard normal values used to approximate the gradients of the expectations found in $\mathcal{J}$.

Note further that the interior of the second summation in $\nabla_L \mathcal{J}(x)$ reduces into the following two cases depending on the value of ϵ_j,

$$
\begin{aligned}
&\nabla_L \min \left\{0, \log p(x + L\epsilon_j) - \log p(x)\right\} \\
&= \begin{cases} 0 & \text{if } \log p(x + L\epsilon_j) \geq \log p(x) \\ \frac{1}{p(x+L\epsilon_j)} p'(x + L\epsilon_j)\epsilon_j^T & \text{if } \log p(x + L\epsilon_j) < \log p(x) \end{cases}.
\end{aligned}
$$

The above gradient characterizes the gradient update rule $L_{t+1} = L_t + \gamma \nabla_L \mathcal{J}(x)$ where t is the time step of gradient ascent and γ is the step size used to maximize $\mathcal{J}(x)$. Here we make the practical note that due to the presence of the $p(x + L\epsilon_j)^{-1}$ term in the gradient, ϵ_j values that result in proposals with negligible density can cause an explosion of the gradient. We thus set a large threshold value of h to catch elements in the gradient matrix with absolute values greater than h, and set the offending values to $\pm h$ respectively. This event is rare in practice but useful in tail geometries where the fraction of potentially offending proposals is higher.

Now, if we use this procedure to identify a value of L to maximize $\mathcal{J}(x)$ for each point x, call these L_x, we could then characterize a Metropolis-Hastings algorithm using these Cholesky factors.

However, we recognize that a great number of steps would be necessary to optimize $\mathcal{J}(x)$ to within some small error threshold. As gradient updates can be computationally expensive, executing a complete run of gradient ascent at every iteration of an MCMC algorithm would be untenable.

We propose that instead of fully optimizing $\mathcal{J}(x)$ at every iteration, a process that requires many expensive steps, we perform one step of gradient ascent at every MCMC iteration. This will provide approximations of the point-wise optimal sampler discussed so far with the following justifications. First, we note that the early steps of gradient ascent tend to be the most influential and thus a complete run of gradient ascent is not absolutely necessary. Secondly, in practical contexts, changes in geometry are typically gradual which implies that nearby points experience similar behaviour, and by extension, similar gradients. While the proposal distribution is not fully optimized at every iteration, on aggregate, the proposal distributions become more optimal as iterations progress.

2.2 Algorithm Details

We now gather the results of the above discussions into a complete algorithm summary. The Divergence Minimization sampler's objective function and gradient update rule produce a series of covariance matrices for generating Gaussian proposals with an MCMC framework. Consistent with the acceptance rule in the objective function, we incorporate a Metropolis rule for proposal acceptance. Plainly, at each iteration, we accept the proposal y from the current position x_t with probability:

$$\alpha(x_t, y|C_t) = \min\left\{1, \frac{p(y)}{p(x_t)}\right\}$$

where t is the current MCMC iteration, and C_t is the current Cholesky factor of the proposal distribution's covariance matrix. Note that C_t represents the partially optimized Cholesky factor as opposed to the fully optimized L_x used previously. We reserve discussion of convergence issues for Section 2.4.

Algorithm 1 summarises the DM sampler with perpetual adaption. In its most basic form, the initial Cholesky factor is set to a diagonal matrix with equal scaling along each dimension though a more complex initialisation would also be valid. Furthermore, parameters including the step size and balancing parameters are constant and supplied as inputs although they can be adapted along with the Cholesky factor.

2.3 Divergence Minimization: A Case Study

To understand the behaviour of the DM sampler, we examine a case study using a single banana distribution. The banana distribution is a unimodal distribution with non-Gaussian contours. For context, the contours of this distribution are presented in Figure 1. The banana distribution is known to be a difficult

Algorithm 1 Divergence Minimization Sampler with Perpetual Adaption

Inputs (defaults): target $p(x)$, balancing parameter β (0.2), initial point x_0, step size γ (0.002), step threshold h ($10/\gamma$), initial scaling σ (2), iterations M
Initialize: $C_0 := \sigma\mathbb{1}$
for t = 0,...,M **do**
    Generate $\epsilon_t \sim N(\mathbf{0}, \mathbb{1})$
    Propose $y = x_t + C_t\epsilon_t$
    Compute $G = \nabla_L \mathcal{J}(x_t)$
    Accept y with probability $\alpha(x_t, y|C_t)$
    Update $x_{t+1} = y$ if accepted or $x_{t+1} = x_t$ if rejected.
    If element $|G_{ij}| > h$, set $G_{ij} = \text{sign}(G) \cdot h$
    Update Cholesky Factor: $C_{t+1} \leftarrow C_t + \gamma G$
end for

distribution to sample from with basic MCMC algorithms because of its quickly changing local geometry, especially in the two tails [Haario and Saksman, 1998, Haario et al., 2001].

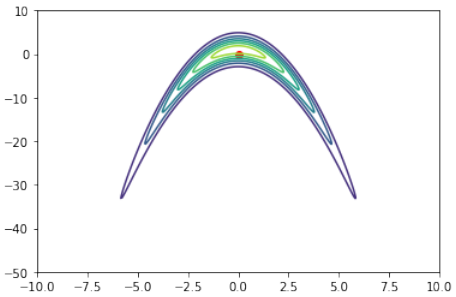
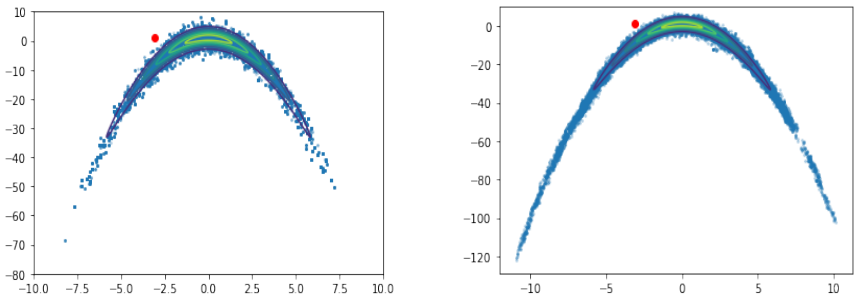


Figure 1: Banana distribution contours. (Note: lighter contours indicate higher density, red dot indicates the origin)

An intuitive way to understand the behaviour of the DM sampler is to examine its samples. Figure 2 presents parallel results from an adaptive Random Walk Metropolis (aRWM) and DM sampler run. The aRWM algorithm used in this case study and subsequent examples is described by Roberts and Rosenthal [2008]. Each algorithm was run for 20,000 iterations with the first 1,000 removed as burn-in. Visually, we notice in the DM sampler results in Figure 2b that the interior of the contours is evenly explored whereas aRWM has blank gaps within the tails of the contours.

Quantitatively, we compare the algorithms using the acceptance rate and the expected squared jumping distance (ESJD). The ESJD used here balances the goals of a high acceptance rate with the increased exploration of larger steps and is defined as $ESJD = \sum_{t=2}^{M} ||x_t - x_{t-1}||_2^2$ [Gelman and Pasarica, 2010, Roberts



(a) aRWM *(b) DM sampler*

Figure 2: Banana distribution samples. The aRWM samples are more sparse and there are gaps in the tails whereas the DM sampler produces more samples in the tails that reach further outwards. (Note: Red dot indicates starting points and blue dots indicate samples)

and Rosenthal, 2001]. The DM sampler produced an acceptance rate of 71.43% with an ESJD of 2.1 as compared to the 8.45% acceptance rate and ESJD of 8.4 of aRWM. Since we know that the DM sampler has a higher acceptance rate, this suggests that the DM sampler takes smaller steps and is perhaps less efficient in terms of exploration than aRWM. With that said, more careful steps suggest a lower risk of missing regions of interest.

Such behaviours can be explained by examining the contours of the proposal distribution at different points of the target distribution. Figure 3 presents the contours of the final proposal distribution of the aRWM run centered at the final sample. In other words, they are the contours of the covariance matrix of all samples generated. Notice that the contours have largely failed to adapt to the specific geometry of the target distribution. They have simply expanded so that all regions of meaningful density in the target are covered by the proposal distribution at any given time but have not conformed to the unique geometry of the target distribution [Bédard, 2007]. One might expect that we could achieve similar success with even a simple RWM algorithm, given a large enough proposal distribution.

We contrast this behaviour to that demonstrated by the DM sampler in Figure 4. The DM sampler delivers on the promise of adaptation to local behaviour as illustrated by the contours closely matching the region of interest. The proposal distributions benefit from adapted covariance matrices that align with the current tail, resulting in a dramatically reduced likelihood of bad proposals as compared to the aRWM proposals.

In summary, what we have observed is the intended behaviour of the DM sampler to adapt to local regions of interest. This manifests in the contour plots. Such behaviour aligns with the objective of producing desirable adaptation. While it is common for algorithms to simply adapt to the *scale* of a target distribution, the DM sampler adapts to the *behaviour* of the target distribution,

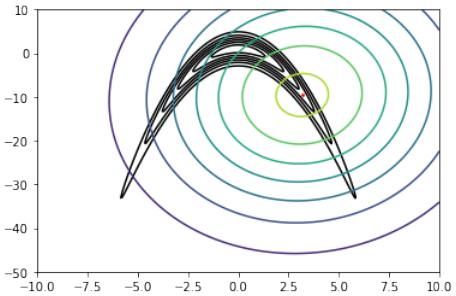
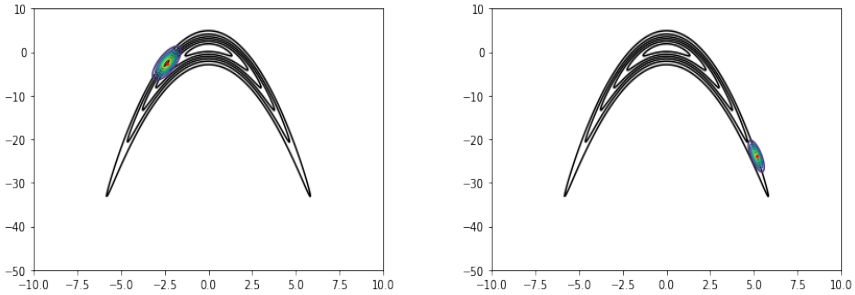


Figure 3: Proposal distribution contours from the final iteration of aRWM centred at the final sample imposed on the banana distribution contours. Notice that the contours do not match the behaviour of the target distribution.



(a) Left tail contour *(b) Right tail contour*

Figure 4: Sample DM sampler contours from the same algorithm execution. Notice that the proposal contours in the given iterations conform to the local geometry of the target distribution.

a completely different and much more challenging task that is especially handy for distributions with unique geometry. Furthermore, while in this instance, it seems that aRWM outperforms in efficiency, we must question whether adapting to just the *scale* of a target distribution is scalable in higher dimensions given that the density will become more and more sparse.

2.4 Convergence and Finite Adaptation

In a standard adaptive scheme, the algorithm typically involves certain technical conditions (such as diminishing adaptation and containment, or finite adaption) to guarantee convergence to the target distribution [Roberts and Rosenthal, 2007, Rosenthal, 2011]. In this work, we argue that certain target distributions, such as those with unusual geometry or those with many unique modes, lend themselves to perpetual adaptation as no single Gaussian proposal distribution could hope to sample well in all regions of interest. The banana example in the

previous section is a good example to illustrate this. As the banana distribution is clearly non-Gaussian, a single non-adapting Gaussian proposal cannot appropriately orient itself in the apex *and* in both tails. However, the consequence of embracing perpetual adaptation is that the standard convergence framework for adaptive MCMC is no longer compatible.

Our goal is to sample efficiently using region-specific proposal distributions while still fulfilling the requirements of the standard convergence framework. As such, we propose a two phase approach that limits adaptation to a finite number of iterations and subsequently transfers the lessons learned in adaptation to a Metropolis-Hastings framework. By limiting adaption to a finite number of iterations, convergence of the non-adaptive phase to the target distribution is guaranteed [Roberts and Rosenthal, 2007, Rosenthal, 2011].

Recall that the basis of the DM sampler is to approximate the optimal proposal distribution characterized by the Cholesky factor L_x that maximizes the objective function $s(x)$. As discussed in Section 2.1, if we knew the values of all L_x, we could produce a simple Metropolis-Hastings algorithm with defined proposal distributions. Of course, we have seen that optimizing $s(x)$ is difficult for a single point, let alone all points in space. Fortunately, the procedure described in Algorithm 1 constructs Cholesky factors C_t at each iteration t to approximate the given location's optimal proposal structure for sampling. If we record these Cholesky factors after each iteration, they can act as a proxy for the optimal Cholesky factor for nearby points as well, assuming some degree of continuity. Thus, after generating a collection of points and their associated Cholesky factors in the adaptive phase, in each iteration of the non-adaptive phase, we select the Cholesky factor associated with the closest adaptive phase sample to construct a proposal covariance matrix for the current iteration. The algorithm thus proposes points from the distribution $q(y|x_t) \sim N(x_t, C_t C_t^T)$ and accept with the following rule:

$$\alpha_f(x_t, y|C_t, \widetilde{C}_y) = \min\left\{1, \frac{p(y)q(x_t|y)}{p(x_t)q(y|x_t)}\right\}$$

where x_t is the current position, y is the proposal, $q(y|x_t) \sim N(x_t, C_t C_t^T)$, $q(x_t|y) \sim N(y, \widetilde{C}_y \widetilde{C}_y^T)$, and C_t and $\widetilde{C}_y$ are the Cholesky factors from the adaptive phase iterations that correspond to the points closest to x_t and y respectively. In other words, instead of calculating a new Cholesky factor for every new point, we select the point from our adaptive phase that is closest to the new point and use its corresponding (approximate) Cholesky factor. This non-adaptive phase adheres to the standard validity criteria of a non-adaptive Metropolis-Hastings algorithm. A complete algorithm summary of this scheme is presented in Algorithm 2.

We test the finite adaptation variant of the DM sampler on the banana distribution presented in Section 2.3. Essentially, once the adaptive phase completes, we consolidate the samples and covariance matrices and then begin the non-adaptive phase at the last adaptive phase sample. In Figure 5, we present 20,000 samples generated from the non-adaptive phase. In addition to the visual indication of the non-adaptive samples covering the relevant portions of

the state space, we note that the acceptance rate is 60.62%, the proportion of samples in the left side of the distribution is 51.7%, and the sample mean of $\begin{bmatrix}-0.27 & -6.73\end{bmatrix}$ is approaching the true mean. These diagnostics indicate the algorithm is sampling well and is converging to the target distribution as expected.

We note that the finite adaption version of the DM sampler performs similarly to the perpetually adapting version with the added benefit of adhering to established convergence criteria.

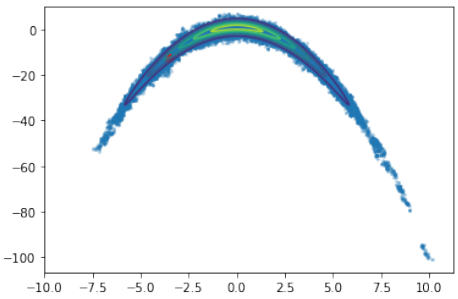


Figure 5: Finite adaptation DM sampler variant. Samples presented only include those from the non-adaptive phase.

Remark: We now comment on the choice of the Metropolis acceptance rule for the original DM sampler as well as discuss an alternative that could perhaps motivate future work. Recall that each iteration of the adaptive phase triggers the gradient update rule. Any proposal under this framework will be asymmetric which at first glance would suggest the use of a Metropolis-Hastings acceptance rule [Hastings, 1970]. Suppose for a moment that we were to consider a Metropolis-Hastings rule. In other words, we replace the acceptance rule α with the following:

$$\alpha^*(x_t, y|C_t) = \min\left\{1, \frac{p(y)q(x_t|y)}{p(x_t)q(y|x_t)}\right\}$$

where x_t is the current position, y is the proposal, C_t is the Cholesky factor of the proposal covariance matrix, $q(y|x_t) \sim N(x_t, C_tC_t^T)$, and $q(x_t|y) \sim N(y, (C_t + \gamma\nabla_L\mathcal{J}(x_t))(C_t + \gamma\nabla_L\mathcal{J}(x_t))^T)$. The distribution of $q(x_t|y)$ considers the gradient step made in the process of moving from x_t to y, reflecting the asymmetry involved in returning from y to x_t. In this case, reversibility would be upheld at each individual iteration without introducing any finite adaptation [Roberts and Smith, 1994, Bai et al., 2011, Craiu et al., 2015]. However, the proposal kernels across iterations are not necessarily identical. Concretely, visiting, leaving, and then returning to a point can result in different proposal kernels at the same point due to the use of only a single gradient step at each iteration. Thus, each individual step under the hypothetical Metropolis-Hastings setup

Algorithm 2 Divergence Minimization Sampler with Finite Adaptation

Inputs (defaults): target $p(x)$, balancing parameter β (0.2), initial point x_0, step size γ (0.002), step threshold h ($10/\gamma$), initial scaling σ (2), iterations M, finite adaptation threshold F ($M/2$), finite subsample size s ($M/20$)
Initialize: $C_0 := \sigma\mathbb{1}$
Adaptive Phase
for t = 0,...,F **do**
    Generate $\epsilon_t \sim N(\mathbf{0}, \mathbb{1})$
    Propose $y = x_t + C_t\epsilon_t$
    Compute $G = \nabla_L \mathcal{J}(x_t)$
    Accept y with probability $\alpha(x_t, y|C_t)$
    Update $x_{t+1} = y$ if accepted or $x_{t+1} = x_t$ if rejected.
    If element $|G_{ij}| > h$, set $G_{ij} = \text{sign}(G) \cdot h$
    Update Cholesky Factor: $C_{t+1} \leftarrow C_t + \gamma G$
end for
Let S be a sample of s points from $0, 1, ...F$
Non-Adaptive Phase
for t = F+1,...,M **do**
    Select $C_t := C_i$ where $i \in S$, such that $d(x_i, x_t)$ is minimized.
    Generate $\epsilon_t \sim N(\mathbf{0}, \mathbb{1})$
    Propose $y_t = x_t + C_t\epsilon_t$
    Select $\widetilde{C}_y := C_j$ where $j \in S$, such that $d(x_j, y)$ is minimized.
    Accept y with probability $\alpha_f(x_t, y|C_t, \widetilde{C}_y)$
    Update $x_{t+1} = y$ if accepted or $x_{t+1} = x_t$ if rejected.
end for

would be reversible, but in aggregate, the entire chain may not be. This perpetual adaption represents a departure from the established convergence theory. In this paper, we have instead decided to proceed with a finite adaptation scheme that does guarantee convergence to the target distribution.

3 Scout MCMC

So far, we have focused on methods to sample from target distributions with irregular geometry. The DM sampler accomplishes this task by focusing on adapting local regions. The adaption is both rapid and effective. While the DM sampler can be executed on any target, if the target has modes separated by large low probability boundaries, it does not have an explicit mechanism to cross these barriers. Now, we introduce an extension of the DM Sampler that attempts to apply this rapid local adaptation strategy to multimodal targets. We term it Scout MCMC.

3.1 Motivation

One of the core challenges of multimodal sampling is the need to sample efficiently from varied local geometries in each mode. As mentioned previously, parallel tempering attempts to solve the problem of identifying modes and crossing low probability boundaries by running multiple tempered chains [Sambridge, 2013]. However, there is no mechanism to sample efficiently once the non-tempered chain is swapped into a new region. Even if some adaptive procedure such as local covariance matrix estimation is introduced after every swap, there is no guarantee that sufficient iterations will have passed to meaningfully adapt before another swap move occurs. By contrast, the key advantage of the DM sampler introduced in this work is that its adaptive mechanism is immediate. The gradient update does not rely on having a bank of good samples to generate a covariance matrix.

Thus, with Scout MCMC, we propose a two chain hybrid of the DM sampler with parallel tempering where the non-tempered chain is simply the DM sampler and the second chain is tempered by either a factor provided by the user or one proportional to the number of dimensions [Tawn et al., 2019]. The rationale behind two chains instead of the typical many chains used in parallel tempering algorithms is that the DM sampler renders low tempered chains unnecessary since it can sufficiently explore regions on its own and only requires heavily tempered chains to help move between modes. Such a feature allows us to overcome one of the primary disadvantages of parallel tempering: the excess computational burden of running many chains.

3.2 Algorithm Details

Scout MCMC generates proposals for the main chain $q(y_t|x_t) \sim N(x_t, C_t C_t^T)$, and accepts with probability:

$$\alpha(x_t, y_t|C_t) = \min\left\{1, \frac{p(y_t)}{p(x_t)}\right\}$$

Then, it adapts the main chain Cholesky factor by $\gamma \nabla_L \mathcal{J}(x_t)$ as before. Next, a proposal for the scout chain is generated as $q(c_t|s_t) \sim N(s_t, \sigma_s \mathbb{1})$, which is accepted according to the following rule:

$$\alpha_s(s_t, c_t) = \min\left\{1, \frac{p(c_t)^\tau}{p(s_t)^\tau}\right\}$$

Finally, Scout MCMC considers swapping x_{t+1} and s_{t+1} every k iterations according to the swap rule:

$$\alpha_{\text{swap}}(x, s) = \min\left\{1, \frac{p(x)^\tau p(s)}{p(x) p(s)^\tau}\right\}$$

Algorithm 3 provides pseudocode for the implementation of Scout MCMC. Once again, control over step size and initial scaling is determined by the user to allow flexibility between targets. For example, depending on the expected global

Algorithm 3 Scout MCMC with Perpetual Adaption

Inputs (defaults): target $p(x)$, balancing parameter β (0.2), temperature τ (0.1), initial point x_0, step size γ (0.002), step threshold h ($10/\gamma$), initial scaling σ (2), tempered scaling σ_s (9), iterations M, swap frequency k (20)
Initialize: $C_0 := \sigma\mathbb{1}$, $s_0 = x_0$
for t = 0,...,M **do**
    Main Chain Step
    Generate $\epsilon_t \sim N(\mathbf{0}, \mathbb{1})$
    Propose $y_t = x_t + C_t\epsilon_t$
    Compute $G = \nabla_L \mathcal{J}(x)$
    Accept y_t with probability $\alpha(x_t, y_t|C_t)$
    Update $x_{t+1} = y_t$ if accepted or $x_{t+1} = x_t$ if rejected.
    If element $|G_{ij}| > h$, set $G_{ij} = \text{sign}(G) \cdot h$
    Update Cholesky Factor: $C_{t+1} \leftarrow C_t + \gamma G$
    Scout Step
    Propose $c_t \sim N(s_t, \sigma_s\mathbb{1})$ and accept with probability $\min\left\{1, \frac{p(c_t)^\tau}{p(s_t)^\tau}\right\}$
    Update $s_{t+1} = c_t$ if accepted or $s_{t+1} = s_t$ if rejected.
    Swap Step
    if $t \equiv 0 \mod k$ **then**
        Swap x_{t+1} and s_{t+1} with probability $\min\left\{1, \frac{p(s_{t+1})p(x_{t+1})^\tau}{p(x_{t+1})p(s_{t+1})^\tau}\right\}$
    end if
end for

region of interest, the tempered chain scaling can be adjusted. Additional details can be added such as adapting the scaling of the tempered chain or varying the limit on the frequency of swap moves.

3.3 Finite Adaptation

Similar to the DM sampler, we present a two-phase finitely adapting variant of Scout MCMC. The first phase is the procedure presented in Algorithm 3. In the second, non-adaptive phase, the structure of the scout chain does not change. However, the main chain follows the same process as the finitely adapting DM sampler where the Cholesky factor corresponding to the nearest iteration of the adapting phase is used to construct proposal distributions in the non-adaptive phase. This reduces the non-adapting phase to a Metropolis-Hastings algorithm. We present the pseudocode associated with the finitely adapting Scout MCMC in Algorithm 4.

In the next section, we will see examples demonstrating that similar to the DM sampler, the finite adaption version of Scout MCMC performs similarly in practice to the perpetually adapting version but has the theoretical advantage of adhering to established convergence criteria.

Algorithm 4 Scout MCMC with Finite Adaptation

Inputs (defaults): target $p(x)$, balancing parameter β (0.2), temperature τ (0.1), initial point x_0, step size γ (0.002), step threshold h ($10/\gamma$), initial scaling σ (2), tempered scaling σ_s (9), iterations M, finite adaptation threshold F ($M/2$), swap frequency k (20), finite subsample size s ($M/20$)
Initialize: $C_0 := \sigma\mathbb{1}$, $s_0 = x_0$
Adaptive Phase
for t = 0,...,F **do**
    Main Chain Step
    Generate $\epsilon_t \sim N(\mathbf{0}, \mathbb{1})$
    Propose $y_t = x_t + C_t\epsilon_t$
    Compute $G = \nabla_L \mathcal{J}(x)$
    Accept y_t with probability $\alpha(x_t, y_t|C_t)$
    Update $x_{t+1} = y_t$ if accepted or $x_{t+1} = x_t$ if rejected.
    If element $|G_{ij}| > h$, set $G_{ij} = \text{sign}(G) \cdot h$
    Update Cholesky Factor: $C_{t+1} \leftarrow C_t + \gamma G$
    Scout Step
    Propose $c_t \sim N(s_t, \sigma_s\mathbb{1})$
    Accept c_t with probability $\min\left\{1, \frac{p(c_t)^\tau}{p(s_t)^\tau}\right\}$
    Update $s_{t+1} = c_t$ if accepted or $s_{t+1} = s_t$ if rejected.
    Swap Step
    if $t \equiv 0 \mod k$ **then**
        Swap x_{t+1} and s_{t+1} with probability $\min\left\{1, \frac{p(s_{t+1})p(x_{t+1})^\tau}{p(x_{t+1})p(s_{t+1})^\tau}\right\}$
    end if
end for
Let S be a sample of s points from $0, 1, ...F$
Non-Adaptive Phase
for t = F+1,...,M **do**
    Main Chain Step
    Select $C_t := C_i$ where $i \in S$, such that $d(x_i, x_t)$ is minimized.
    Generate $\epsilon_t \sim N(\mathbf{0}, \mathbb{1})$
    Propose $y_t = x_t + C_t\epsilon_t$
    Select $\widetilde{C}_t := C_j$ where $j \in S$, such that $d(x_j, y)$ is minimized.
    Accept y with probability $\alpha_f(x_t, y|C_t, \widetilde{C}_t)$
    Update $x_{t+1} = y$ if accepted or $x_{t+1} = x_t$ if rejected.
    Scout Step
    Propose $c_t \sim N(s_t, \sigma_s\mathbb{1})$ and accept with probability $\min\left\{1, \frac{p(c_t)^\tau}{p(s_t)^\tau}\right\}$
    Update $s_{t+1} = c_t$ if accepted or $s_{t+1} = s_t$ if rejected.
    Swap Step
    if $t \equiv 0 \mod k$ **then**
        Swap x_{t+1} and s_{t+1} with probability $\min\left\{1, \frac{p(s_{t+1})p(x_{t+1})^\tau}{p(x_{t+1})p(s_{t+1})^\tau}\right\}$
    end if
end for

4 Examples

In this section, we examine the performance of the DM Sampler and Scout MCMC using a variety of target distributions. We focus on multimodal distributions with an emphasis on those with atypical geometry. It is important to note that traditional diagnostics such as effective sample size (ESS) will be misleading in the case of multimodal distributions [Turner and Neal, 2017, Elvira et al., 2018]. ESS specifically may prefer a sample that fails to leave the initial mode as compared to a sample that explores modes separated by a low probability chasm. ESJD is arguably a better diagnostic as it increases with increased step size and acceptance rate, both being favourable behaviours. Recall that the ESJD is defined as $ESJD = \sum_{t=2}^{M} ||x_t - x_{t-1}||_2^2$.

Given that there is a lack of consensus on appropriate diagnostics for targets with more than one mode, we have selected target distributions with easily computed true expected values to use as reference points for the simulations. Going forward, we will refer to the true expected value as $E[X]$, the estimated expected value with an MCMC sample as $\hat{E}[X]$, and the Euclidean distance between the true and estimated values as $d(E[X], \hat{E}[X])$.

As an example of a target distribution with easily computed expectations, the basis vector target that will be discussed in detail consists of a mixture of Gaussian distributions where each component Gaussian lies on one of the basis vectors and all are equidistant from the origin. This leads to a target with negligible density at the origin but with an expected value that is simply at the origin itself. A similar but much more challenging target consisting of a mixture of banana distributions presents a target with a mean at the origin that also has complex geometry. In these instances, we can use the distance from the sample mean to the origin, the true mean, to evaluate algorithm performance.

In the following examples, we compare the DM Sampler and Scout MCMC with standard Random Walk Metropolis (RWM), adaptive RWM (aRWM), and Parallel Tempering (PT). For clarity, RWM generates proposals with a single shared Gaussian distribution, aRWM generates proposals using the empirical covariance matrix up to the current iteration, and PT executes multiple RWM chains on the target distribution with different levels of tempering applied [Swendsen and Wang, 1986, Geyer, 1991]. For consistency, we match the maximum tempering level used by parallel tempering to the level used by the scout chain in Scout MCMC. We also execute two versions of parallel tempering: one with 2 chains to match Scout MCMC, and one with 5 chains as would be more likely in practice. Finally, we include both the fully adaptive versions of the DM Sampler and Scout MCMC along with the variants that limit adaptation and transition to a second non-adaptive phase. The code used to generate the following examples along with implementations of each algorithm in Python are provided to supplement the discussion[1].

[1] https://github.com/AmeerD/Scout-MCMC

4.1 Double Banana Distribution

The first distribution we consider is an extension of the banana distribution examined in Section 2.3. Specifically, we consider a pair of banana distributions with overlap in the tails. This results in two primary modes along the curves of the two bananas as well as two secondary modes at the intersections. Figure 6 provides the contours of this distribution.

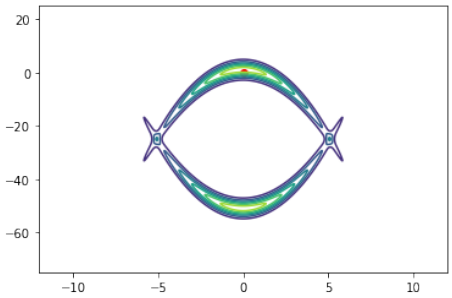


Figure 6: Double banana distribution contours. (Note: lighter contours indicate higher density, red dot indicates the origin)

All of the algorithms are run for 50,000 iterations with the first 1,000 samples discarded as burn-in. Table 1 presents the results of this experiment. For this specific distribution, the target mean is $\begin{bmatrix} 0 & -25 \end{bmatrix}$.

Notice that the samples of aRWM and Scout MCMC have gotten closest to the mean of the distribution. It is worth noting that there is negligible density at the mean as illustrated by Figure 6 so the ability to achieve the correct mean indicates that both bananas have been visited. In comparison, standard RWM and the parallel tempering have not achieved the level of success of the other algorithms. Surprisingly, even with 5 chains, parallel tempering has largely failed to converge within the 50,000 iterations. The distinction between aRWM and Scout MCMC lies in the efficiency diagnostics. We see that Scout MCMC accepts over 10 times as many proposals as aRWM though it has a smaller ESJD. A large acceptance rate is not necessarily indicative of a better algorithm but with both algorithms performing similarly, this could indicate that Scout MCMC produces higher quality proposals. Since ESJD is a measure of both the acceptance rate and the step size made with each move but Scout MCMC has a much higher acceptance rate, this would indicate that aRWM proposes moves with much greater step sizes than Scout MCMC. This is expected, however, as Scout MCMC uses a user-specified cooldown period where the main chain makes local moves and does not swap with the scout chain. Finally, we note that the finite adaptation variants of the DM sampler and Scout MCMC both perform similarly to their fully adapting counterparts though they tend to accept fewer proposals.

	Accept (%)	$\hat{E}$[X]		d(E[X],$\hat{E}$[X])	ESJD
RWM	51.89	+0.31	−17.77	7.24	0.78
aRWM	7.76	+0.16	−24.84	0.23	40.2
PT (2 chains)	37.59	−2.29	−19.31	6.13	0.99
PT (5 chains)	40.25	−2.40	−4.210	20.92	2.16
DM Sampler	83.50	−2.35	−23.76	2.66	0.80
DM Finite	68.59	−2.25	−24.02	2.45	0.61
Scout MCMC	83.41	−0.22	−23.78	1.24	11.1
Scout Finite	73.57	−0.22	−18.83	6.17	11.8

Table 1: Double banana target results. We see here that aRWM and Scout MCMC produced sample means that are closest to the true mean. While aRWM has a greater ESJD value, Scout MCMC has a greater acceptance rate.

In addition to sample diagnostics, we also examine the samples themselves in Figure 7. A notable observation is the dramatic imbalance of the parallel tempering samples in Figures 7c and 7d as well as the DM samples in Figure 7e. RWM also experiences slight imbalance but more notably does not reach far into the tails within the number of iterations. The contrast between the DM sampler and Scout MCMC samples highlights the regulating abilities of the Scout chain to help the DM sampler escape from extreme regions. In this example, the left tail in Figure 7e could be considered an extreme region. Both the aRWM and Scout MCMC plots exhibit desirable sampling behaviour. The samples are well dispersed over the target and seemingly balanced. The primary difference is the relative concentration due to aRWM tending to reject proposals and stay at the same points whereas Scout MCMC produces a larger number of unique points.

Finally, we plot the samples generated by the finite versions of the DM sampler and Scout MCMC in Figure 8. The samples presented largely match those of the fully adaptive versions. This indicates that the bank of covariance matrices generated in the adaptive phase is sufficient to produce region specific samples as intended.

4.2 Basis Vector Distribution

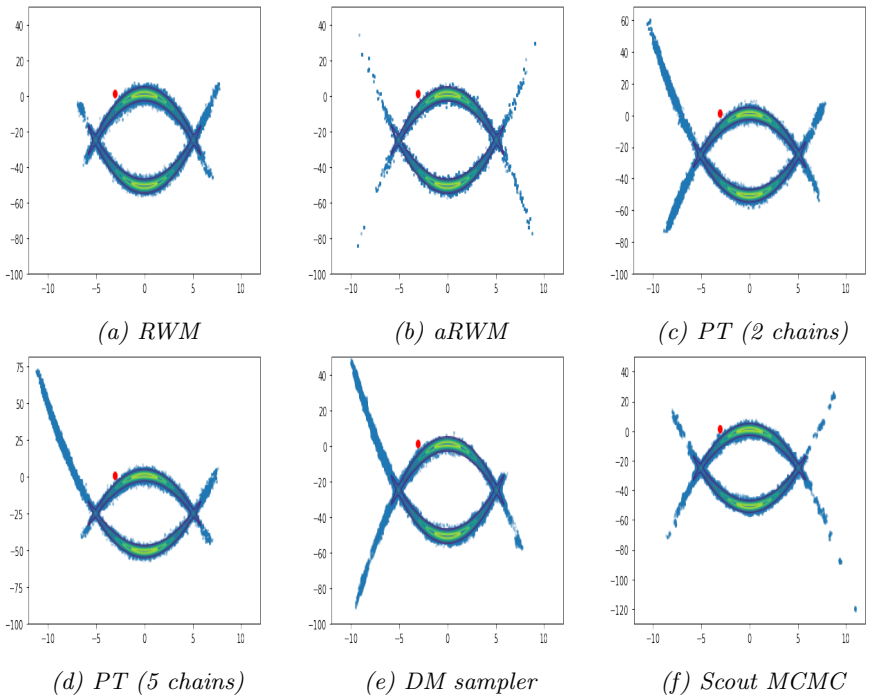
As described briefly in the prelude to this section, the basis vector target consists of a series of normal distributions along the basis vectors in $\mathbb{R}^4$. Formally, the distribution is a mixture of the following normal distributions:

$$N\left(10\mathbf{e}_1, \mathbf{I}_4\right), N\left(-10\mathbf{e}_1, \mathbf{I}_4\right), N\left(10\mathbf{e}_2, \mathbf{I}_4\right), N\left(-10\mathbf{e}_2, \mathbf{I}_4\right),$$

$$N\left(10\mathbf{e}_3, \mathbf{I}_4\right), N\left(-10\mathbf{e}_3, \mathbf{I}_4\right), N\left(10\mathbf{e}_4, \mathbf{I}_4\right), \text{and } N\left(-10\mathbf{e}_4, \mathbf{I}_4\right)$$

where $\mathbf{e}_i$ is a basis vector in the ith direction and $\mathbf{I}_4$ is the 4D identity matrix.

The key feature of this distribution is that the expected value is at the origin but there is negligible density at the origin. As such, any MCMC algorithm that hopes to be successful must be able to cross a vast low probability desert to move between modes. Each algorithm was run for 40,000 iterations with the first 2,000



(a) RWM *(b) aRWM* *(c) PT (2 chains)*

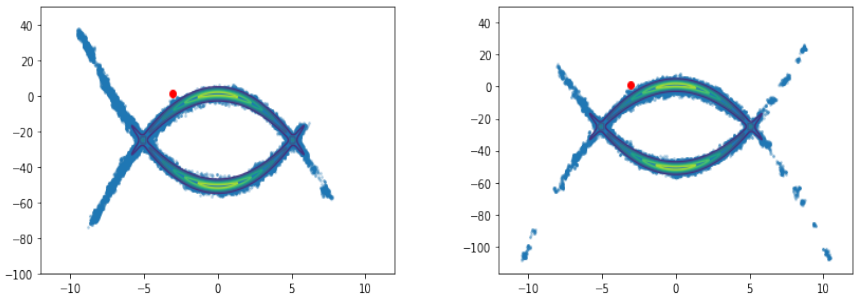
(d) PT (5 chains) *(e) DM sampler* *(f) Scout MCMC*

Figure 7: Double Banana Samples. PT and the DM sampler have trouble moving away from tail regions. The best performing algorithms are aRWM and Scout MCMC as they achieve the most accurate sample mean and highest ESJD values. (Note: Red dot indicates starting points and blue dots indicate samples)

iterations discarded as burn-in. The results of the 4D basis vector target are presented in Table 2.

Note that all of RWM, aRWM, and the DM sampler produced sample means that were quite far from the origin, which is the true mean. This indicates that these algorithms did not visit all of the modes in a balanced manner and got stuck in a mode. This behaviour is not unexpected, however, as these algorithms have no mechanism to cross low probability boundaries. Scout MCMC and parallel tempering, in contrast, produced sample means approaching the origin with Scout MCMC outperforming both cases of parallel tempering. In this instance, there does not appear to be any major impact from increasing the number of chains from two to five in parallel tempering aside from a larger ESJD. Finally, we note that the finite variants of the DM sampler and Scout MCMC perform in line with their respective fully adapting versions, validating their specification as non-adapting approximations.

To confirm that all eight modes were visited (as opposed to, say, two opposing modes with a mean that is equal to the origin), we examine the trace plots of parallel tempering with 5 chains and Scout MCMC in Figure 9. Each trace plot



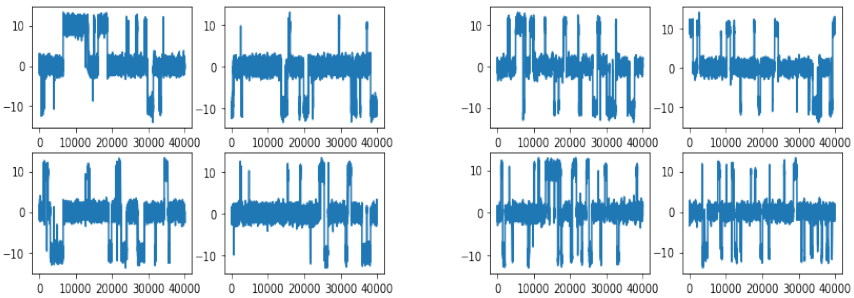
(a) DM Sampler *(b) Scout MCMC*

Figure 8: Double Banana Samples from the finite variants of the DM Sampler and Scout MCMC. Both algorithms variants perform in line with their perpetually adapting counterparts.

represents one of the dimensions and for this target distribution, we should see the trace plots reaching -10, 0 and 10 in all dimensions. From the plots, it is clear that both algorithms are capable of moving between modes in a frequent manner and that all modes have been visited. It is at this juncture that we turn to the point of efficiency. Notice the difference in acceptance rate and ESJD between Scout MCMC and parallel tempering in Table 2. Scout MCMC has a tendency to accept almost twice as many proposals as parallel tempering even though parallel tempering takes larger steps. This is in part due to parallel tempering having no limit on the frequency of swap moves whereas Scout MCMC is set to only be able to consider swapping every 20 iterations.

	Accept (%)	$d(\mathrm{E}[\mathrm{X}],\hat{E}[\mathrm{X}])$	ESJD
RWM	37.88	10.03	1.09
aRWM	29.43	10.03	1.09
PT (2 chains)	37.63	2.67	1.47
PT (5 chains)	37.38	2.76	2.13
DM Sampler	70.89	10.09	0.39
DM Finite	69.72	10.05	0.39
Scout MCMC	70.60	1.01	1.01
Scout Finite	71.06	1.26	1.04

Table 2: 4D basis vector target results. Both versions of Scout MCMC and PT produced sample means that were closest to the true mean. PT slightly outperforms Scout MCMC in terms of ESJD but both variants of Scout MCMC have slightly closer means and higher acceptance rates.



(a) PT (5 chains) *(b) Scout MCMC*

Figure 9: 4D basis vector trace plots. Notice that both PT and Scout MCMC are able to reach -10, 0, and 10 in all four dimensions.

4.3 Banana Bunch Distribution

The final and most challenging target consists of a mixture of 12 banana distributions in $\mathbb{R}^3$ arranged such that there is even less interaction than there is in the double banana example. We call this distribution the banana bunch. As this example is in $\mathbb{R}^3$, we cannot simply present the contours. However, the distribution can be understood as the mixture of three groups. The projection of the target on each pair of axes (x-y, x-z, and y-z) appears as the contours in Figure 10.

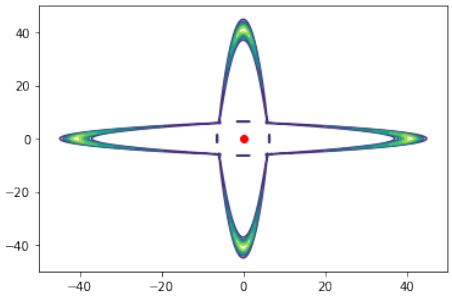


Figure 10: Projections of the banana bunch distribution on each pair of axes (x-y, x-z, and y-z). (Note: lighter contours indicate higher density, red dot indicates the origin)

Combining all three groups will result in the intersection of the apexes of two component distributions at ± 40 along each axis. This results in a target with six modes, each far from the origin, and 24 tails extending from the modes towards each other. Once again, we capitalize on the symmetry of our targets and find the expected value to be at the origin. Though the origin has negligible density, scatter plots of samples projected down to planes composed of the basis vectors can be slightly misleading. Figure 11 presents a sample generated directly from

the target distribution. The points seemingly at the origin are actually "above" and "below" the origin with respect to the axis missing from the respective plot.

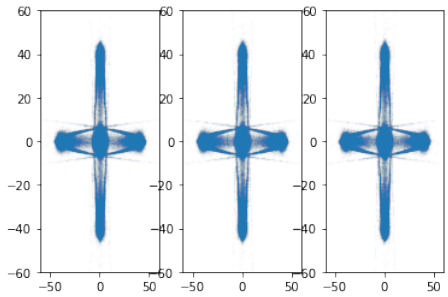


Figure 11: True banana bunch samples. Presented are samples generated directly from the target distribution projected onto the x-y, x-z, and y-z plane.

Given the more complex nature of this distribution, we increase the number of samples for all algorithms to 100,000 with the first 1,000 discarded as burn-in. In addition to acceptance rate, first moment, and ESJD, we also consider the second moment, the expectation of the element-wise square of the samples. Such an expectation will assess how well the tails have been explored. A sample that concentrates too heavily in the 6 modes will overshoot this expectation even if it successfully produces a mean near the origin. The true value of this expectation is $\begin{bmatrix}400 & 400 & 400\end{bmatrix}$. Table 3 presents the results of our experiment.

	Accept (%)	$d(E[\mathrm{X}],\hat{E}[\mathrm{X}])$	$d(E[\mathrm{X}^2],\hat{E}[\mathrm{X}^2])$	ESJD
RWM	48.19	17.04	342.9	1.19
aRWM	1.76	1.97	338.0	14.00
PT (2 chains)	41.38	15.01	377.7	1.72
PT (5 chains)	42.88	11.76	132.0	3.54
DM Sampler	77.05	3.83	316.7	0.98
DM Finite	57.71	12.05	334.0	0.77
Scout MCMC	79.11	1.26	88.5	13.53
Scout Finite	63.34	2.6	109.6	10.99

Table 3: Banana bunch results. Although most algorithms come quite close to the true expected mean, not all are successful in finding the squared mean. The best performing ones are Scout MCMC, the finite variance of Scout MCMC, and PT with five chains. However, even with five chains, PT has a much less favourable ESJD than either Scout MCMC, each only utilizing two chains.

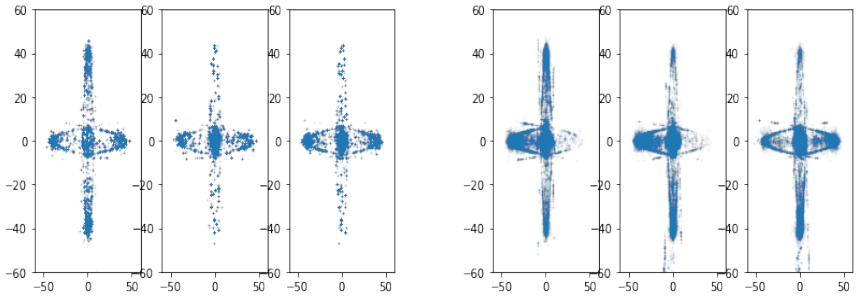
Perhaps the most surprising result in Table 3 is that the sample mean produced by each algorithm is quite close to the true expected value although

aRWM and Scout MCMC are clearly the best performers on that front. Regarding the squared expected value, Scout MCMC performs the best even though the results for parallel tempering with 5 chains presents a convincing case for its convergence properties. We note however that parallel tempering with 2 chains fails to match the performance of either Scout MCMC or parallel tempering with 5 chains. This suggests that 2 chains is not generally sufficient without a more nuanced strategy on the main chain such as using the DM sampler in Scout MCMC. Finally, with respect to ESJD, we note that aRWM and Scout MCMC are clearly the best performers using this efficiency metric. However, we warn the reader to recognize that aRWM accepted less than 2% of proposals, did not produce a sample second moment that was close to the true second moment, and finished the algorithm with a covariance matrix with diagonal $\begin{bmatrix}862 & 1000 & 209\end{bmatrix}$. This is an indication that as the dimension increases, aRWM relies heavily on expanding its reach to cover the whole region of interest rather than conforming to the shape of the target distribution. The cost of this behaviour is that the proposals are not always of high quality and one must hope that it produces enough proposals to extract a decent set of good samples in a limited amount of time. In this case, the enormous proposal distribution was not sufficient to fully explore the tails leading in from the modes towards the origin, thus producing an inaccurate sample second moment.

In order to be complete, however, we must also examine the plots of samples (projected to 2D) to understand whether our algorithms truly explore all modes and tails. Once again, we refer the reader to Figure 11 to view the expected behaviour. Given that RWM and the DM sampler have no mechanisms for inter-mode movement and that the numerical results reflect this fact, we focus on the remaining algorithms from here. We also focus on the 5 chain version of parallel tempering given that it outperformed the 2 chain version.

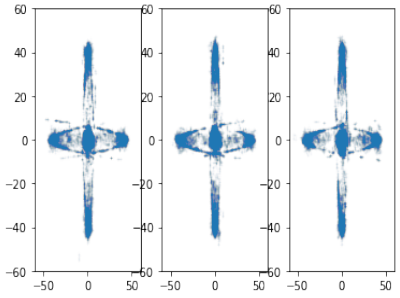
The visual results of Figure 12 reflect the behaviours noted by the diagnostics in Table 3. Adaptive RWM struggles to explore the tails as well as the z-axis which results in the poor squared expected value. Parallel tempering, in contrast, performs better in the tails but does not explore the distribution evenly within the specified number of iterations which manifests in the poorer expected value. In addition, there are a number of samples well beyond the modes that are the result of a swap from a tempered chain to the main chain. With Scout MCMC and its finite variant, we see the most appropriate distinction between mode and tail concentrations which manifests in the best squared expectation. They also do so with a higher frequency of points than aRWM or parallel tempering. The highly efficient proposals are realized even when the large distance between modes of the basis vector example is combined with the unusual geometry of the banana examples thus illustrating the ability of Scout MCMC to deliver on the promises of a multimodal sampler that excels at rapid adaptation to local geometry.

We conclude this section by repeating the notion of efficient and effective sampling. We find that aRWM may be efficient from an ESJD perspective and parallel tempering is effective as a way to explore different modes, but neither prove to be adequate across the board. Instead, Scout MCMC proves itself as

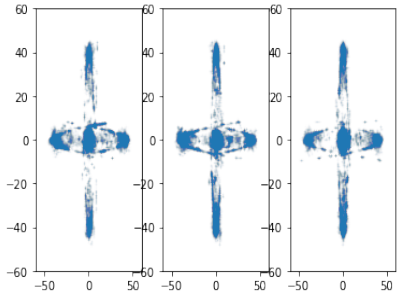


(a) Adaptive RWM

(b) Parallel Tempering (5 chains)



(c) Scout MCMC



(d) Scout Finite

Figure 12: Banana Bunch Samples (projected onto the x-y, x-z, and y-z planes). aRWM produces much sparser plots that reflect its inability to explore the tails of the bananas. PT, despite having more samples, still results in inaccurate sample statistics because of its imbalance, with many points skewing to one side. The two Scout MCMC variants produce the most optimal results with a large and balanced set of samples as well as strong exploration of the banana tails.

an efficient and effective "smart" sampler by adopting a strategy of combining rapid regional adaptation with heavy tempering for mode swapping.

5 Discussion and Future Work

In this paper we have introduced an algorithm designed to rapidly adapt to the local behaviour of a given target distribution. Such adaptation is accomplished through the minimization of the information projection of the KL Divergence between the target distribution and the proposal distribution family. By combining this Divergence Minimization sampler with one highly tempered chain, we leverage the positive mode-finding capabilities exhibited by parallel tempering algorithms while retaining the ability to quickly explore modes with unique and complex geometries. Our framework offers an approach to multimodal sampling with few assumptions where both the issues of mode finding and local exploration are accomplished during the execution of the algorithm. Finally, we

leverage the adaptation of the DM sampler and Scout MCMC in a two-stage algorithm that uses the produced covariance matrices of the DM sampler and Scout MCMC in the first phase to initialize a follow up Metropolis-Hastings phase that adheres to standard convergence criteria. This finite adaptation algorithm continues to use optimized local samplers to efficiently sample from local geometries without needing perpetual adaptive steps.

We have presented the baseline algorithm in this paper and believe that there is much room for future research. For example, the criteria required for a non-diminishing perpetually adaptive algorithm to converge remains underexplored. Moreover, one might also be interested in studying whether there is an optimal frequency to adaptation and swapping or whether there are certain target geometries that are more or less challenging to explore. Smaller changes such as adapting step size and other fixed parameter inputs are also possibilities. Finally, there is certainly room to explore different objective functions in the DM sampler. Some possibilities include testing alternate similarity measures or replacing the regulating term to encourage different behaviours. Such modifications could further improve the performance of the DM sampler and Scout MCMC beyond what has been demonstrated in this paper.

A Appendix

A.1 Approximating the Gradient

Since,

$$\mathcal{J}(x) = \beta H_q + \beta E_\epsilon \left[\log p(x + L\epsilon)\right] + E_\epsilon \left[\min\left\{0, \log p(x + L\epsilon) - \log p(x)\right\}\right]$$

the gradient of $\mathcal{J}(x)$ is:

$$\begin{aligned}
&\nabla_L \mathcal{J}(x) \\
&= \nabla_L \beta H_q + \nabla_L \beta E_\epsilon \left[\log p(x + L\epsilon)\right] + \nabla_L E_\epsilon \left[\min\left\{0, \log p(x + L\epsilon) - \log p(x)\right\}\right] \\
&= \beta \nabla_L H_q + \beta E_\epsilon \left[\nabla_L \log p(x + L\epsilon)\right] + E_\epsilon \left[\nabla_L \min\left\{0, \log p(x + L\epsilon) - \log p(x)\right\}\right]
\end{aligned}$$

Consider each of the three terms of the above of $\nabla_L \mathcal{J}(x)$ individually:

Term 1: $\beta \nabla_L H_q$. We can use the form of the entropy of a multivariate normal distribution to evaluate this gradient:

$$\begin{aligned}
\beta \nabla_L H_q &= \beta \nabla_L \left(\frac{k}{2} \log(2\pi e) + \frac{1}{2} \log(|L||L^T|) \right) \\
&= \beta \nabla_L \left(\frac{k}{2} \log(2\pi e) + \frac{1}{2} \sum_{i=1}^{k} \log L_{ii}^2 \right) \\
&= \beta \nabla_L \left(\sum_{i=1}^{k} \log L_{ii} \right) \\
&= \beta \text{diag} \left(\frac{1}{L_{11}}, \ldots, \frac{1}{L_{kk}} \right)
\end{aligned}$$

Term 2: $\beta E_\epsilon \left[\nabla_L \log p(x + L\epsilon)\right]$. First, note the following:

$$\beta E_\epsilon \left[\nabla_L \log p(x + L\epsilon)\right] = \beta E_\epsilon \left[\frac{1}{p(x + L\epsilon)} p'(x + L\epsilon) \epsilon^T \right]$$

The expectation on the right-hand side does not simplify cleanly. However, the interior of the expectation is simple enough to evaluate for a given ϵ. As such we can draw a number of $\epsilon_j \sim N(0, 1)$ at each iteration and compute an unbiased estimate of $\beta E_\epsilon \left[\nabla_L \log p(x + L\epsilon)\right]$ with Simple Monte Carlo. That is, at each iteration, we compute:

$$\beta E_\epsilon \left[\nabla_L \log p(x + L\epsilon)\right] \approx \frac{1}{J} \sum_{j=1}^{J} \frac{\beta}{p(x + L\epsilon_j)} p'(x + L\epsilon_j) \epsilon_j^T$$

Term 3: $E_\epsilon \left[\nabla_L \min\left\{0, \log p(x + L\epsilon) - \log p(x)\right\}\right]$. Similar to the second piece of the gradient, note that this expectation does not simplify but we can

produce an unbiased estimate by relying on a series of draws of $\epsilon_j \sim N(0,1)$ in a given iteration.

$$\begin{aligned} &E_\epsilon \left[\nabla_L \min\left\{0, \log p(x + L\epsilon) - \log p(x)\right\}\right] \\ &\approx \frac{1}{J}\sum_{j=1}^{J} \nabla_L \min\left\{0, \log p(x + L\epsilon_j) - \log p(x)\right\} \end{aligned}$$

However, the presence of the minimum operator suggests this summation will not simplify in the same way as the previous component. Considering the two cases, we can naturally separate them depending on if $\log p(x + L\epsilon_t) \geq \log p(x)$.

In the first case, if $\log p(x + L\epsilon_t) \geq \log p(x)$, acceptance of the proposal under a Metropolis framework is guaranteed and:

$$\nabla_L \min\left\{0, \log p(x + L\epsilon_t) - \log p(x)\right\} = 0$$

If $\log p(x + L\epsilon_t) < \log p(x)$, then the Metropolis ratio is less than 1 and we have,

$$\begin{aligned} &\nabla_L \min\left\{0, \log p(x + L\epsilon_t) - \log p(x)\right\} \\ &= \nabla_L \left(\log p(x + L\epsilon_t) - \log p(x)\right) \\ &= \frac{1}{p(x + L\epsilon_t)} p'(x + L\epsilon_t)\epsilon_t^T \end{aligned}$$

Consolidating the three terms, to search for an optimal local proposal distribution, at each iteration of the MCMC chain we perform the following gradient-based update (we omit the iteration subscript t on x and L for clarity purposes):

$L_{t+1} = L_t + \gamma \nabla_L \mathcal{J}(x)$ where

$$\begin{aligned} \nabla_L \mathcal{J}(x) = \beta \text{diag}\left(\frac{1}{L_{11}}, \ldots, \frac{1}{L_{kk}}\right) + \sum_{j=1}^{J} \frac{\beta}{p(x + L\epsilon_j)} p'(x + L\epsilon_j)\epsilon_j^T + \\ \sum_{j=1}^{J} \nabla_L \min\left\{0, \log p(x + L\epsilon_j) - \log p(x)\right\}. \end{aligned}$$

Note further that the interior of the Term 3 summation reduces into the following two cases depending on the value of ϵ_j,

$$\begin{aligned} &\nabla_L \min\left\{0, \log p(x + L\epsilon_j) - \log p(x)\right\} \\ &= \begin{cases} 0 & \text{if } \log p(x + L\epsilon_j) \geq \log p(x) \\ \frac{1}{p(x+L\epsilon_j)} p'(x + L\epsilon_j)\epsilon_j^T & \text{if } \log p(x + L\epsilon_j) < \log p(x) \end{cases}. \end{aligned}$$

Note that the current position is denoted x, the proposal is $y = x + L\epsilon$, the standard multivariate draw is ϵ_j, and γ is the predetermined step size.

References

C. Andrieu and J. Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18, 12 2008. doi: 10.1007/s11222-008-9110-y.

Y. Atchadé, G. Fort, E. Moulines, and P. Priouret. Adaptive Markov Chain Monte Carlo: Theory and Methods. *Bayesian Time Series Models*, 01 2011. doi: 10.1017/CBO9780511984679.003.

Y. Bai, G. O. Roberts, and J. S. Rosenthal. On the Containment Condition for Adaptive Markov Chain Monte Carlo Algorithms. *Adv. Appl. Stat.*, 21, 01 2011.

S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

M. Bédard. Weak convergence of Metropolis algorithms for non-i.i.d. target distributions. *The Annals of Applied Probability*, 17(4):1222–1244, Aug 2007. ISSN 1050-5164. doi: 10.1214/105051607000000096. URL `http://dx.doi.org/10.1214/105051607000000096`.

R. V. Craiu, J. S. Rosenthal, and C. Yang. Learn From Thy Neighbor: Parallel-Chain and Regional Adaptive MCMC. *Journal of the American Statistical Association*, 104(488):1454–1466, 2009. doi: 10.1198/jasa.2009.tm08393. URL `https://doi.org/10.1198/jasa.2009.tm08393`.

R. V. Craiu, L. Gray, K. Łatuszyński, N. Madras, G. O. Roberts, and J. S. Rosenthal. Stability of adversarial Markov chains, with an application to adaptive MCMC algorithms. *The Annals of Applied Probability*, 25(6): 3592–3623, Dec 2015. ISSN 1050-5164. doi: 10.1214/14-aap1083. URL `http://dx.doi.org/10.1214/14-AAP1083`.

V. Elvira, L. Martino, and C. P. Robert. Rethinking the Effective Sample Size, 2018.

A. Gelman and C. Pasarica. Adaptively Scaling the Metropolis Algorithm Using Expected Squared Jumped Distance. *Statistica Sinica*, 20, 01 2010. doi: 10.2139/ssrn.1010403.

C. J. Geyer. Markov chain Monte Carlo maximum likelihood. *Computing Science and Statistics*, 1991.

H. Haario and E. Saksman. Adaptive Proposal Distribution for Random Walk Metropolis Algorithm. *Computational Statistics*, 14, 07 1998. doi: 10.1007/s001800050022.

H. Haario, E. Saksman, and J. Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242, 04 2001. URL `https://projecteuclid.org:443/euclid.bj/1080222083`.

W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970. ISSN 00063444. URL http://www.jstor.org/stable/2334940.

N. Higham. Cholesky Factorization. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1:251 – 254, 09 2009. doi: 10.1002/wics.18.

K. Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012. ISBN 9780262018029.

E. Pompe, C. Holmes, and K. Łatuszyński. A Framework for Adaptive MCMC Targeting Multimodal Distributions, 2019.

G. O. Roberts and J. S. Rosenthal. Optimal Scaling for Various Metropolis-Hastings Algorithms. *Statistical Science*, 16, 11 2001. doi: 10.1214/ss/1015346320.

G. O. Roberts and J. S. Rosenthal. Coupling and Ergodicity of Adaptive Markov Chain Monte Carlo Algorithms. *Journal of Applied Probability*, 44(2):458–475, 2007. doi: 10.1239/jap/1183667414.

G. O. Roberts and J. S. Rosenthal. Examples of Adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367, 2008. doi: 10.1198/jcgs.2009.06134. URL https://doi.org/10.1198/jcgs.2009.06134.

G. O. Roberts and A. F. M. Smith. Simple conditions for the convergence of the Gibbs sampler and Metropolis-Hastings algorithms. *Stochastic Processes and their Applications*, 49(2):207 – 216, 1994. ISSN 0304-4149. doi: https://doi.org/10.1016/0304-4149(94)90134-1. URL http://www.sciencedirect.com/science/article/pii/0304414994901341.

J. S. Rosenthal. *Optimal Proposal Distributions and Adaptive MCMC*, chapter 4. CRC Press, 2011.

T. Salimans, D. P. Kingma, and M. Welling. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap, 2015.

M. Sambridge. A Parallel Tempering algorithm for probabilistic sampling and multimodal optimization. *Geophysical Journal International*, 196(1):357–374, 10 2013. ISSN 0956-540X. doi: 10.1093/gji/ggt342. URL https://doi.org/10.1093/gji/ggt342.

C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948. URL http://dblp.uni-trier.de/db/journals/bstj/bstj27.html#Shannon48.

R. H. Swendsen and J.-S. Wang. Replica Monte Carlo Simulation of Spin-Glasses. *Phys. Rev. Lett.*, 57:2607–2609, Nov 1986. doi: 10.1103/PhysRevLett.57.2607. URL https://link.aps.org/doi/10.1103/PhysRevLett.57.2607.

N. G. Tawn, G. O. Roberts, and J. S. Rosenthal. Weight-Preserving Simulated Tempering, 2019.

M. Titsias and P. Dellaportas. Gradient-based Adaptive Markov Chain Monte Carlo. In *Advances in Neural Information Processing Systems*, volume 32, pages 15730–15739. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/624567140fecc40163fed3c45a959a7c-Paper.pdf`.

R. Turner and B. Neal. How well does your sampler really work?, 2017.

T. Yamano. A generalization of the Kullback-Leibler divergence and its properties. *Journal of Mathematical Physics*, 50, 03 2009. doi: 10.1063/1.3116115.