METRON (2022) 80:343–370
https://doi.org/10.1007/s40300-022-00235-y

Plateau proposal distributions for adaptive component-wise multiple-try metropolis

F. Din-Houn Lau[1] · Sebastian Krumscheid[2]

Received: 15 June 2021 / Accepted: 18 June 2022 / Published online: 15 July 2022

Abstract
Markov chain Monte Carlo (MCMC) methods are sampling methods that have become a commonly used tool in statistics, for example to perform Monte Carlo integration. As a consequence of the increase in computational power, many variations of MCMC methods exist for generating samples from arbitrary, possibly complex, target distributions. The performance of an MCMC method, in particular that of a Metropolis–Hastings MCMC method, is predominately governed by the choice of the so-called proposal distribution used. In this paper, we introduce a new type of proposal distribution for the use in Metropolis–Hastings MCMC methods that operates component-wise and with multiple trials per iteration. Specifically, the novel class of proposal distributions, called *Plateau* distributions, does not overlap, thus ensuring that the multiple trials are drawn from different regions of the state space. Furthermore, the Plateau proposal distributions allow for a bespoke adaptation procedure that lends itself to a Markov chain with efficient problem dependent state space exploration and favourable burn-in properties. Simulation studies show that our novel MCMC algorithm outperforms competitors when sampling from distributions with a complex shape, highly correlated components or multiple modes.

Keywords Component-wise Metropolis–Hastings · Multiple-try Metropolis · Adaptive Markov chain Monte Carlo · Plateau proposal distribution

1 Introduction

Markov chain Monte Carlo (MCMC) methods are essentially used to perform Monte Carlo integration, which has become a standard statistical tool. Specifically, MCMC methods produce samples from a target distribution π by constructing an ergodic Markov chain whose stationary distribution is π. Typically, MCMC methods are used when it is difficult to sample

✉ Sebastian Krumscheid
krumscheid@uq.rwth-aachen.de

F. Din-Houn Lau
dhlpaper1@gmail.com

1 Cheshunt, Hertfordshire, UK

2 Department of Mathematics, RWTH Aachen University, 52062 Aachen, Germany

from the target distribution directly, e.g. when the normalisation constant is unknown. There are many ways to construct this Markov chain, which have led to many variations of MCMC methods; see, e.g., [2].

The classic MCMC method is the Metropolis-Hastings algorithm [21]. At each iteration, the Metropolis-Hastings algorithm is designed to update the entire current state (i.e., all components of the random vector generated at the previous iteration) at once. However, updating individual components, or subsets of components, is possible. Indeed, this type of component-wise updating was initially proposed in [21], but did not receive much attention at first. Recently, component-wise updates in MCMC methods have been more established in the literature; see, e.g., [14] where the convergence rates of such methods are studied. In this paper, we focus on updating individual components. Using individual component updates is a way of sampling from the (lower dimensional) conditional distributions of the target, provided the conditional distributions are known. However, the conditional distributions are unknown in practise. Modelling the conditional distributions using parametric models leads to an inflexible approximation, whereas non-parametric models do not scale well with the number of dimensions. For these reasons, we focus our attention on independent component-wise updates.

Another variant of MCMC sampling, separate from component-wise updating, is the multiple-try method [15], where several proposals or *trials* are suggested at each iteration. The motivation behind the multiple-try method is that more of the space is explored at the expense of an increased computational cost (i.e., proposal generation and evaluation of acceptance criterion). Several variations of multiple-try MCMC algorithms exist in the literature; see [20].

Recently, in [27] the authors introduce a component-wise, multiple-try MCMC method with Gaussian proposals for each trial, where each univariate proposal has the same mean but different variances. In this work, we introduce a new class of proposal distributions for use in a component-wise, multiple-try MCMC method. These proposals, called *Plateau* distributions, do not overlap to exploit the multiple-try nature of the method. Indeed, by using proposals that do not overlap for each trial, the Markov chain is forced to explore different parts of the state space. Conversely, using proposals that overlap, such as Gaussians with the same mean and different variances, can lead to an inefficient algorithm, as the trials tend to be from a similar region of the state space.

The idea of Plateau proposals is intuitive, easy to implement and leads to good results, in the sense of exploring the state space. Moreover, using the Plateau proposals leads to a reversible Markov chain with the target distribution as its invariant distribution, see, e.g., [27].

As is common for most proposals used in MCMC methods, the Plateau distributions depend on parameters that need to be selected with respect to the target distribution to obtain an effective algorithm. Adaptation of MCMC methods typically entails *tuning* the parameter(s) of a class of proposal distributions, e.g., the variance parameter in a Normal distribution, in order to improve the convergence properties of the Markov chain. For instance, in [11], a Gaussian proposal distribution is used whose variance (or covariance) is adapted using the previously generated states of the Markov chain.

In this work, we propose an adaptation procedure that is designed for use with the non-overlapping Plateau proposals. The Plateau proposals together with the associated adaptation procedure are illustrated in Fig. 1. Suppose the MCMC algorithm is initiated with multiple-try proposals whose distributions are presented by the different coloured lines in Fig. 1a. These proposals operate independently on each (univariate) component in the target space. As the Markov chain evolves, the number of times each Plateau's proposed candidate state is selected

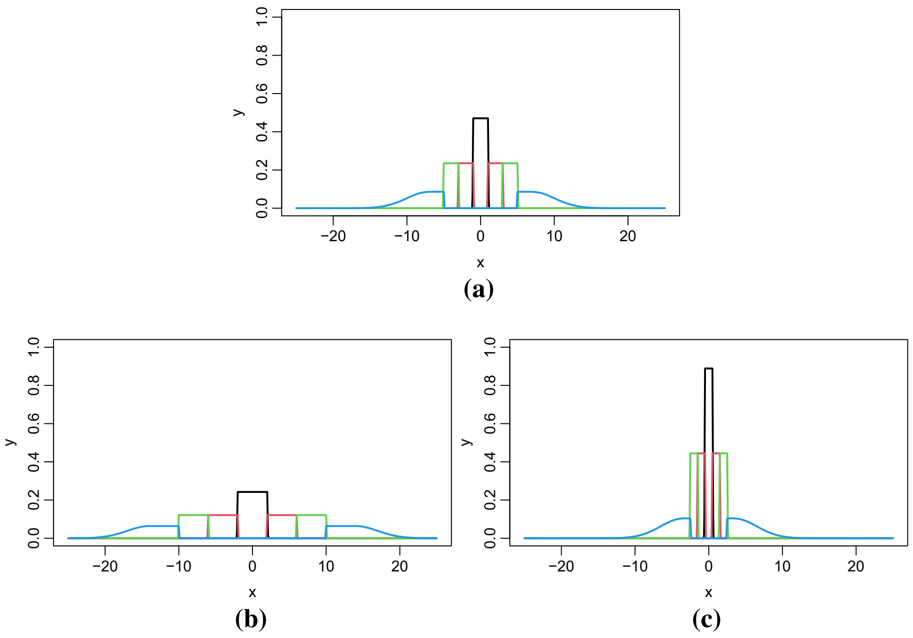


Fig. 1 Illustration of 4 Plateau proposal distributions (probability density functions) represented by the different colours and the adaptation procedure. The initial Plateau distributions are presented in (**a**); If the outermost (blue) proposals are selected frequently, the proposals are adapted to (**b**); If the innermost (black) proposals are selected frequently, the proposals are adapted to (**c**) (colour figure online)

is recorded. If the innermost or outermost Plateaus are overly selected, then the set of Plateau proposals are re-organised as depicted in Figs. 1b and 1c. More precisely, if the innermost (black) proposal is overly selected, then one halves the width of each plateau and shifts the set of proposals to remove the gaps. A similar procedure is performed if the outermost (blue) proposals is overly selected by doubling the width of the plateaus. This procedure appropriately scales the set of Plateau proposals to the components of the target distribution. This intuitive adaptation procedure makes explicit use of the non-overlapping feature of the Plateau proposals. Specifically, if a sample is selected from a particular Plateau distribution, then that sample could not have been obtained by sampling from any other Plateau distribution since their supports do not overlap. The advantages of using non-overlapping proposals, as opposed to overlapping proposals, is explored in Sects. 5 and 7. The mathematical definition of the Plateau proposals and the pseudocode for the adaptation procedure are presented in Sects. 3 and 4 respectively.

The works in [9] and [5] share a similar objective with our proposed work, namely: draw samples that are far apart from each other to facilitate an effective exploration of the state space. For example, in [9] the authors introduce an independent (i.e., component-wise), yet single-try, Metropolis–Hastings method using a Normal mixture distribution as proposal. The mixture distribution is adapted using a k-means clustering approach in order to explore the space more efficiently. In [5] an improved state space exploration is achieved by combining variance reduction techniques with multiple trials. Specifically, it involves drawing multiple trails from a single, well-chosen proposal distribution that is based on, e.g., Latin Hypercube sampling. The performance of the approaches in [9] and [5] depends on a well-chosen transformation. Conversely, the approach introduced in this work does not require such a transformation, as the Plateau proposals reside in the state space. Further, our adaptation procedure tunes the Plateau proposals parameters as the MCMC runs and does not require additional clustering (i.e., optimisation) algorithms to be performed.

To summarise, the setting of this work is to propose a new MCMC algorithm along with an adaptation procedure, which is simple to implement, intuitive and can be used without using explicit derivative information (e.g., Hamiltonian Monte Carlo [22]) or conditional distributions (e.g., Gibbs sampling [4]) of the target. Therefore, a general-purpose MCMC algorithm is required. To this end, we introduce Plateau proposals as a general class of proposal distributions for use in a component-wise multiple-try MCMC methods. The combination of the non-overlapping characteristic of Plateau proposals with the multiple-try approach and a bespoke adaptation procedure leads to good results for a variety of target distributions. Examples of target distributions that will benefit from an adaptive multiple-try MCMC based on Plateau proposals include, in particular, multimodal distributions, such as mixture distributions, and, more generally, target distributions with complex shapes, such as those with highly correlated components or Bayesian posteriors.

The remainder of this paper is organised as follows. In Sect. 2.1 a generic component-wise multiple-try algorithm is presented. The novel class of Plateau proposals is introduced and discussed in Sect. 3. In Sect. 4 we discuss how to adaptively select the parameter of the Plateau proposals and offer a detailed algorithmic description of the complete method. The performance of our new method is then compared with other MCMC methods in Sect. 6. Finally, a commentary on improvements and a summary are provided in Sect. 8.

2 Component-wise update with multiple trials

Let π be a probability density function, $\pi : \mathcal{X} \to \mathbb{R}^+$, where $\mathcal{X} \subseteq \mathbb{R}^d$. Our main interest is to sample from π; this is the target distribution. We assume that sampling directly from π is difficult or impossible, for example because π may only be known up to a multiplicative constant. In order to sample from π we use MCMC methods. In the Metropolis-Hastings algorithm, one of the simplest MCMC methods, a candidate $\boldsymbol{Y}$ for the chain's next state $\boldsymbol{X}_{n+1}$ is drawn from the probability density function (PDF) $T(\boldsymbol{x}, \cdot) : \mathcal{X} \to \mathbb{R}^+$, based on the current state $\boldsymbol{X}_n = \boldsymbol{x} \in \mathcal{X}$ of the Markov chain. The proposal $T(\boldsymbol{x}, \cdot)$ is typically easier to sample from than the target distribution. The density $T(\boldsymbol{x}, \cdot)$ is the conditional density given the current value $\boldsymbol{x}$. This density is commonly known as the proposal distribution. For example, the random-walk proposal [12, 21] uses a multivariate Normal distribution, which we will write as $T(\boldsymbol{x}, \cdot) = \mathrm{N}(\boldsymbol{x}, \sigma^2 I_d)$ with $\sigma > 0$ given and I_d being the $d \times d$ identity matrix. Another example is $\mathrm{N}\big(\boldsymbol{x} + \tau \nabla \ln\big(\pi(\boldsymbol{x})\big), 2\tau I_d\big)$ with $\tau > 0$ fixed, which is the proposal used in the Metropolis-adjusted Langevin algorithm [25]. The realisation of the Markov chain's next state is then selected according to an accept-reject procedure designed such that the resulting Markov chain's stationary distribution is π.

Constructing an MCMC method that produces good results typically depends on the choice of proposal distribution. In particular, the choice of the proposal may significantly affect the properties of the MCMC method, including the speed of convergence to equilibrium and mixing properties [26]. Typically, the proposal distribution is selected from some family of distributions, e.g., from the family of Normal distributions. The performance of MCMC methods depends on appropriately scaling of the proposal distribution, especially for high-dimensional target distributions [23].

Instead of proposing a single multivariate candidate from $\boldsymbol{Y} \sim T(\boldsymbol{x}, \cdot)$ by updating all components of the current state $\boldsymbol{X}_n = \boldsymbol{x}$ simultaneously via the (global) proposal distribution $T(\boldsymbol{x}, \cdot)$, it is also possible to split the state space $\mathcal{X}$ into its individual components (or small groups of components) and propose candidates for each component (or group of components) independently. This local (or projected) approach is intuitive, computationally efficient, and reduces the problem of selecting a multidimensional proposal into lower dimensional proposals that are easier to handle. MCMC methods using these types of updates are called component-wise MCMC methods, which may use a potentially different proposal distribution per component of the state [8, Ch. 1]. In this work, we focus on one-dimensional, component-wise proposals.

Considering independent one-dimensional, component-wise proposals is equivalent to the proposal distribution $T(\boldsymbol{x}, \cdot)$ given $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ being separable, in the sense that

$$T(\boldsymbol{x}, \boldsymbol{y}) \equiv T_1(x_1, y_1) \times \cdots \times T_d(x_d, y_d) = \prod_{k=1}^{d} T_k(x_k, y_k) , \tag{1}$$

for any $\boldsymbol{y} = (y_1, \ldots, y_d) \in \mathbb{R}^d$. Here, $T_k(x, \cdot) \colon \mathbb{R} \to \mathbb{R}^+, k = 1, \ldots, d$ with $x \in \mathbb{R}$, denotes the one-dimensional proposal density given x used to draw the candidate for component k. That is, the proposed (global) candidate $\boldsymbol{Y} = (Y_1, \ldots, Y_d) \in \mathbb{R}^d$ is obtained by sampling each $Y_k \sim T_k(x_k, \cdot)$ mutually independent of any other Y_ℓ.

By construction proposing candidates component-wise does not account for correlations between the components. Consequently, the component-wise proposed candidates $\boldsymbol{Y}$ may not be good representatives of the target distribution π (i.e., most candidates $\boldsymbol{Y}$ will be rejected, resulting in a very low acceptance rate), if π has highly correlated components. Therefore, these candidates with independently sampled components may lead to a poor state space exploration and thus to a poor performance of the MCMC method. To mitigate this issue, we will combine component-wise proposals with multiple trials. The multiple-try technique proposes many candidates from a proposal distribution, amongst which the "best" one is selected. Each trial may be proposed from a different proposal. Thus, in combination with component-wise proposals, we generate M independent candidates (or M trials) for each separate component $k = 1, \ldots, d$. Let $T_{j,k}(x, \cdot) \colon \mathbb{R} \to \mathbb{R}^+$, $j = 1, \ldots, M$, denote the proposal PDF of the jth trial for the kth component given $x_k = x$.

2.1 Generic procedure of component-wise multiple-try metropolis

A generic component-wise multiple-try algorithm is now described – the full pseudocode for generating N samples from the target distribution π is presented in Algorithm 1. Each iteration within the MCMC algorithm draws multiple trials and then performs an acceptance-rejection step for each component sequentially.

We now describe the intuition behind the steps of the procedure. Suppose that the state of the chain at the beginning of the nth iteration is $\boldsymbol{x} = (x_1, \ldots, x_d)$. For the first component (i.e., $k = 1$), M independent trials, $z_1, \ldots, z_M \in \mathbb{R}$, are drawn from $T_{j,1}(x_1, \cdot)$, $j = 1, \ldots, M$. These trials are then weighted according to

$$w_{j,1}(z_j, \boldsymbol{x}) = \pi\big((z_j; \boldsymbol{x}_{[-1]})\big) T_{j,1}(x_1, z_j) \lambda_{j,1}(x_1, z_j) ,$$

where $(z; \boldsymbol{x}_{[-i]}) \in \mathbb{R}^d$ denotes the vector that is identical to $\boldsymbol{x}$ except for its ith component which is replaced by $z \in \mathbb{R}$, that is $(z; \boldsymbol{x}_{[-i]}) = (x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_d)$.

Algorithm 1: Generic Component-wise Multiple-Try Metropolis

```
Input: number of trials $M$; number of MCMC realisations $N$; starting position $\boldsymbol{x}_0 \in \mathbb{R}^d$; target
    distribution $\pi$ (possibly un-normalised); proposal distributions $T_{j,k}$
1: Let $\boldsymbol{X}_0 = \boldsymbol{x}_0 = \boldsymbol{x}$.
2: for $n = 1, \dots, N$ do
3:   for $k = 1, \dots, d$ do
4:     Propose $M$ trials: $z_j \sim T_{j,k}(x_k, \cdot)$ for $j = 1, \dots, M$.
5:     Compute the trial weights
           $w_{j,k}(z_j, \boldsymbol{x}) = \pi\big((z_j; \boldsymbol{x}_{[-k]})\big) T_{j,k}(x_k, z_j) \lambda_{j,k}(x_k, z_j)\ , \quad j = 1, \dots, M.$
6:     Draw $y \in \{z_1, \dots, z_M\}$ randomly with probability proportional to $w_{1,k}, \dots, w_{M,k}$.
7:     Draw $x_j^* \sim T_{j,k}(y, \cdot)$ for $j = 1, \dots, M-1$ and let $x_M^* = x_k$.
8:     Let $\boldsymbol{y} = (y; \boldsymbol{x}_{[-k]})$ and compute
           $\alpha = \min\left\{1, \frac{w_{1,k}(z_1, \boldsymbol{x}) + \cdots + w_{M,k}(z_M, \boldsymbol{x})}{w_{1,k}(x_1^*, \boldsymbol{y}) + \cdots + w_{M,k}(x_M^*, \boldsymbol{y})}\right\}.$
       Draw $r \sim \text{Uniform}(0, 1)$.
9:     if $r < \alpha$ then
10:      Accept $\boldsymbol{X}_n = \boldsymbol{y} = (y; \boldsymbol{x}_{[-k]})$ and set $\boldsymbol{x} = \boldsymbol{y}$.
11:    else
12:      $\boldsymbol{X}_n = \boldsymbol{x}$.
13:    end if
14:  end for
15: end for
16: return $\boldsymbol{X}_1, \dots, \boldsymbol{X}_N$
```

The functions $\lambda_{j,k}(x, y)$ with $x, y \in \mathbb{R}$ for any $k = 1, \dots, d$, are non-negative, symmetric functions in x and y which are selected by the user. It is required that $\lambda_{j,k}(x, y) > 0$ whenever $T_{j,k}(x, y) > 0$. Each trial z_j, $j = 1, \dots, M$, has an associated weight $w_{j,1}(z_j, \boldsymbol{x})$. A candidate for the first component of the chain's next state is then randomly selected amongst all trials z_j ($j = 1, \dots, M$) according to these weights. The selected candidate is then accepted or rejected. The remaining components $k = 2, 3, \dots, d$ of the chain's state are updated in order in a similar fashion; Algorithm 1 details the full pseudocode of the MCMC method.

In step 5 in Algorithm 1, the accepted candidate for the kth component is selected randomly from M trails $z_1, \dots, z_M$ with weights proportional to

$$w_{j,k}(z_j, \boldsymbol{x}) = \pi\big((z_j; \boldsymbol{x}_{[-k]})\big) T_{j,k}(x_k, z_j) \lambda_{j,k}(x_k, z_j), \quad j = 1, \dots, M.$$

The weight depends on the function $\lambda_{j,k}$, for which there are a number of choices in the multiple-try literature [15], such as

$$\left(\frac{T_{j,k}(x, y) + T_{j,k}(y, x)}{2}\right)^{-1}, \quad \big\{T_{j,k}(x, y) T_{j,k}(y, x)\big\}^{-\beta} \quad \text{and} \quad 1.$$

In work [27] the authors suggest using

$$\lambda_{j,k}(x, y) = T_{j,k}(x, y) \, \|y - x\|^{\alpha}\ , \tag{2}$$

where $\alpha = 2.9$ based upon a simulation study focusing on large moves in the state spaces. For our proposal algorithm we use (2) with $\lambda_{j,k}$. In simulations, not reported here, we found that

$\alpha = 2.5$ performed best in terms of the mean squared error for a variety of target distributions. Consequently, we use $\lambda_{j,k}$ in (2) with $\alpha = 2.5$ for the remainder of the paper unless stated otherwise. We note that it is beyond the scope of this paper to propose a particular form for the class of functions $\lambda_{j,k}$ due to the problem dependent nature of this choice. Instead, we advocate that users perform a trial run with a variety of $\lambda_{j,k}$ to determine which is best suit for their application and performance metric.

3 Non-overlapping proposal distributions

In principle, any family of proposals $T_{j,k}$ can be used in Algorithm 1. However, a careful choice of the proposals can lead to a more efficient algorithm.

Recall that $T_{j,k}$ is the jth trial proposal distribution for the kth component. The motivation of using multiple trials is to explore a larger region of the state space than is achieved by using a single proposal. Therefore, it would not be beneficial to sample trials (from $T_{1,k}, \dots, T_{M,k}$) that are similar. To illustrate this point, suppose that for a fixed component k we use the proposal distributions $T_{j,k}(x, \cdot) = \mathrm{N}(x, (j\sigma)^2)$ for $j = 1, \dots, M = 5$ with known $\sigma > 0$ and take $x = 0$ without loss of generality. The probability density functions of these proposals with $\sigma = 1$ are presented in Fig. 2.

As illustrated in Fig. 2, these proposals are very similar. Indeed, 99% of $T_{1,k}$'s density mass lies within the interval

$$\mathfrak{J} = \left(-\sigma\Phi^{-1}(0.995), \sigma\Phi^{-1}(0.995)\right) ,$$

where Φ^{-1} is the inverse of the cumulative distribution function of a standard Normal so that $\Phi^{-1}(0.995) \approx 2.6$. For a candidate from the second proposal distribution $Y_{2,k} \sim T_{2,k}(0, \cdot)$, we then have $\mathbb{P}(Y_{2,k} \in \mathfrak{J}) \approx 0.8$ for any $\sigma > 0$. That is, draws from $T_{1,k}$ and $T_{2,k}$ will be located in the same region with high probability. Similar arguments hold for the wider Gaussian proposals. Thus, draws from these Gaussian proposals will tend to be similar, thus leading to an inefficient use of the multiple-try technique. To avoid proposing candidates from similar regions, we seek densities which do not overlap (or overlap to a small degree).

Specifically, we advocate using proposals of the type illustrated in Fig. 3b. That is, each trial for each component-wise proposal distribution combines uniform distributions with exponentially decaying tails. The amount of overlap between different proposals is controlled

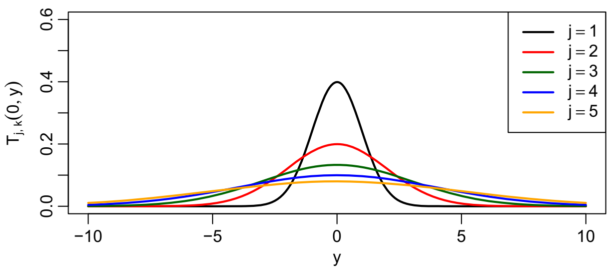
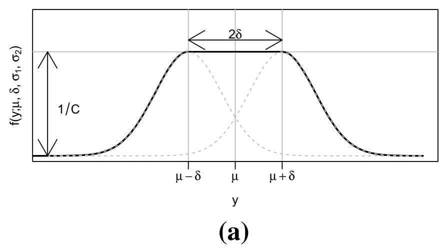


Fig. 2 Probability density function of 5 Normal distributions with zero mean and standard deviations $j\sigma$ for $\sigma = 1$, $j = 1, \dots, 5$



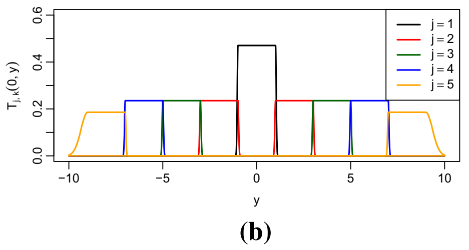


Fig. 3 PDF of Plateau proposal distributions with $M = 5$. The five different proposals each have a different colour

through how fast the tails decay. Further, note that there is no gap between two contiguous distributions, i.e., no interval where neither distribution is likely to be sampled from.

The family of Plateau proposals are now defined. We begin by introducing the PDF (see Fig. 3a)

$$f(y; \mu, \delta, \sigma_1, \sigma_2) = \frac{1}{C} \begin{cases} \exp\left\{-\frac{1}{2\sigma_1^2}\left[y - (\mu - \delta)\right]^2\right\} & \text{for } y < \mu - \delta \\ 1 & \text{for } \mu - \delta \le y \le \mu + \delta \\ \exp\left\{-\frac{1}{2\sigma_2^2}\left[y - (\mu + \delta)\right]^2\right\} & \text{for } y > \mu + \delta \end{cases} \tag{3}$$

where

$$C = \frac{\sqrt{2\pi\sigma_1^2}}{2} + \frac{\sqrt{2\pi\sigma_2^2}}{2} + 2\delta$$

denotes the normalisation constant. For each component k with given value $x_k = x$, we then set the PDF of each trial proposal as

$$T_{j,k}(x, y) = \begin{cases} f(y; x, \delta_1, \sigma, \sigma) & j = 1 \\ \frac{1}{2} f(y; x - 2(j-1)\delta_1 - \delta, \delta, \sigma, \sigma) + \frac{1}{2} f(y; x + 2(j-1)\delta_1 + \delta, \delta, \sigma, \sigma) & j = 2, \ldots, M-1, \\ \frac{1}{2} f(y; x - 2(M-1)\delta_1 - \delta, \delta, \sigma_0, \sigma) + \frac{1}{2} f(y; x + 2(M-1)\delta_1 + \delta, \delta, \sigma, \sigma_1) & j = M \end{cases}$$

for some values of $\delta_1, \delta, \sigma, \sigma_0, \sigma_1 > 0$. The $M = 5$ trial proposals shown in Fig. 3b correspond to $\delta_1 = \delta = 1$, $\sigma = 0.05$ and $\sigma_0 = \sigma_1 = 0.5$. We shall refer to the proposals of this type as *Plateau* proposals, given the shape of their PDFs. The δ_1 parameter controls the width of the central Plateau centred at the current state x. The δ parameter is the width of the other Plateaus. The σ value controls the decay of the tails either side of the inner Plateaus. The outer tails for the Mth proposal are described by σ_0 and σ_1.

To compare with the earlier calculations for coverage probabilities for the Gaussian proposals; 99% of the density of $T_{1,k}$ with $x = 0$, $\delta = 1$, and $\sigma = 0.5$ lies in the interval $\mathfrak{J} = (-2.11, 2.11)$. Suppose that $Y_{2,k} \sim T_{2,k}(0, \cdot)$, then $\mathbb{P}(Y_{2,k} \in \mathfrak{J}) \approx 0.43$, which is reduced by almost a factor of two compared to the overlapping Gaussian proposal. Further, if $\sigma = 0.25$ then $\mathbb{P}(Y_{2,k} \in \mathfrak{J}) \approx 0.31$ and if $\sigma = 0.05$ then $\mathbb{P}(Y_{2,k} \in \mathfrak{J}) \approx 0.06$. Thus, the Plateau proposals overlap less than the Gaussian proposals and further, the extent of the overlapping of the proposals is controlled by the values of σ, σ_1, and σ_2.

Note that each Plateau proposal distribution has a support on $\mathbb{R}$. This is to ensure that the support of the target distribution is included within the support of the proposals. In theory, this allows the Markov chain to explore the entire support of the target distribution. In a practical setting, however, by selecting the value of σ appropriately, the tails of the distribution decay to zero very quickly, making the inner proposals effectively uniform distributions in view of numerical simulations. Therefore, in practice, one could elect to sample directly from Uniform distributions to increase computational speed. However, in all the experiments performed in this paper, we sample from the exact Plateau proposal distributions defined above.

A diverse set of proposal shapes and sizes are possible through the selection of parameter values $\delta, \delta_1, \sigma, \sigma_0$ and σ_1. In the remainder of the paper, we set $\sigma_0 = \sigma_1 = \varsigma$ for some value of ς, so that the decay of the tails of the outermost proposal, $T_{M,k}$, are the same. Further, we fix the half-width of the proposals to be the same, i.e., $\delta = \delta_1 = \Upsilon$. Fixing these parameters means that the Plateau proposals are defined by 3 parameters: $\Upsilon(= \delta = \delta_1)$, σ, and $\varsigma(= \sigma_0 = \sigma_1)$. The values for these parameters will determine the movement of the Markov chain and thus its performance with respect to a particular target distribution. In principle, one could set the parameter values in an ad-hoc manner, e.g., a manual search over the parameter space until a certain acceptance rate is achieved. A practical approach is to automatically *tune* the parameters as the algorithm runs. Section 4 introduces an adaptation procedure that tunes the proposal parameter, $\Upsilon(= \delta = \delta_1)$. The remaining σ and $\varsigma(= \sigma_0 = \sigma_1)$ parameters need to be selected by the user. In simulations, we use $\sigma = 0.05$ and $\varsigma = 3$ so that there is minimal overlap of the Plateau proposals and to ensure the outermost proposal has heavy tails. Results using $\sigma = 0.05$ and $\varsigma = 3$ led to a good performance for a variety of target distributions.

In summary, Plateau proposals is a set of distributions that do not overlap (or overlap very little) and have no gaps between contiguous distributions. Using non-overlapping proposals within multiple-try MCMC methods means that the multiple tries explore different regions efficiently. In addition, no gaps ensure that areas between trial distributions are likely to be explored. The notion of overlapping and gaps of the Plateau proposals are explored in detail in Sect. 7.

4 Adaptation of plateau proposals

As is typically the case when working with MCMC methods, the parameters of the proposal distributions need to be appropriately tuned for the algorithm to be effective. Instead of manually tuning the parameters, an automated method can be used to adapt the proposals as the MCMC procedure runs. These adaptive methods use the information revealed at each step in the MCMC algorithm to tune the parameters of the proposals. For instance, [11] proposes updating the covariance of a multivariate Normal proposal using an empirical estimate of the covariance of the target. In the following, we discuss an adaptation mechanism for use with the Plateau proposals introduced in Sect. 3. The adaptation mechanism is designed to maintain the non-overlapping and gap features of the Plateau proposals. In fact, combining the multiple-try paradigm with the localised shape of the proposals offers a natural adaptation criterion by monitoring preferred (component-wise) proposals, which will allow sampling from the state space in a more structured way.

In the innermost for-loop in Algorithm 1 (steps 4 to 8) with Plateau proposals $T_{j,k}$, only one trial is selected (step 6). The selected trial is generated from a specific non-overlapping Plateau. Over the Markov chain's iterations, counting which Plateau distributions are selected may offer additional insight into the state space exploration. We advocate a procedure to update the Plateau proposals to avoid two undesirable scenarios: (i) when the innermost proposal is selected too often; and (ii) when the outermost proposal is selected too often. Scenario (i) suggests that the proposal distributions are too wide, such that trials are regularly being suggested near the previous state of the chain, i.e., the majority of the suggested moves

Algorithm 2: Adaptation of MCMC

```
Input: thresholds $\eta_1, \eta_2 > 0$; proposal parameter $\Upsilon = \delta = \delta_1$; iteration number $n$; adaptation interval
    length $L$
1:  if $n = 1$ then
2:    Set $c^n_{1,k} = c^n_{M,k} = 0$ for all $k = 1, \dots, d$
3:  end if
4:  Draw $r \sim \text{Uniform}(0, 1)$
5:  if $\left[r < P_n := \max(0.99^{n-1}, 1/\sqrt{n})\right]$ and $[(n \mod L) = 0]$ then
6:    if $c^n_{1,k} > L\eta_1$ then
7:      Update: $\Upsilon \leftarrow 0.5\Upsilon$
8:    end if
9:    if $c^n_{M,k} > L\eta_2$ then
10:     Update: $\Upsilon \leftarrow 2\Upsilon$
11:   end if
12:   Reset $c^n_{1,k} = c^n_{M,k} = 0$.
13: end if
```

are occurring in the interval $(x - \delta, x + \delta)$, when the current state's component is $x_k = x$. Conversely, scenario (ii) suggests that the proposal distributions are too narrow, such that the trials are regularly being suggested in the "tails", far away from the current position x.

We introduce the following adaptation of the Plateau proposals $T_{j,k}$ to counteract these scenarios. As mentioned in Sect. 3, this adaptation procedure will change the half-width of all the Plateau proposals; namely, the $\Upsilon = \delta = \delta_1$ parameter will be updated. First, adaptation can take place at regular, predefined iteration intervals of length L. Within these intervals, each proposal is selected a number of times. Let $c^n_{j,k}$ denote the number of times $T_{j,k}$ was selected by the nth MCMC iteration.

For scenario (i), if $c^n_{1,k} > L\eta_1$ for some $\eta_1 \in (0, 1)$, then the width of all the Plateaus is halved and the proposals are shifted toward x to leave no gaps between contiguous distributions, i.e., the Plateau proposal parameters are updated as: $\Upsilon \leftarrow 0.5\Upsilon$.

For scenario (ii), if $c^n_{M,k} > L\eta_2$ for some $\eta_2 \in (0, 1)$, then the Plateaus widths are doubled and the proposals are shifted away from $x_k = x$ to leave no gaps between contiguous distributions, i.e., the Plateau proposal parameters are updated as: $\Upsilon \leftarrow 2\Upsilon$.

The proposed adaptation, summarised in Algorithm 2, can be inserted between steps 2 and 3 of the MCMC Algorithm 1. Note that the adaptation operation is performed every L iterations. At iteration n, the adaptation is performed with probability $P_n = \max\left\{0.99^{n-1}, 1/\sqrt{n}\right\}$. This ensures that the amount of adaptation reduces the longer the algorithm runs and thus satisfies the diminishing adaptation condition; see, e.g., [24]. Satisfying the diminishing adaptation condition ensures convergence of the algorithm – see Appendix 1.

In summary, we advocate the use of the adaptation procedure, outlined in Algorithm 2, for our Plateau proposal MCMC. The adaptation procedure updates the $\delta = \delta_1$ parameter of the proposals. In simulations, we initialise these parameters at $\delta = \delta_1 = 1$. As mentioned in Sect. 3, the other proposal parameters are set (and fixed) at $\sigma = 0.05$ and $\varsigma = 3$.

5 Investigation of adaptation of MCMC methods

Before assessing the long-term performance of the adaptive Plateau proposal MCMC in simulations, we first provide further insight into the effects of the adaptation procedure. To

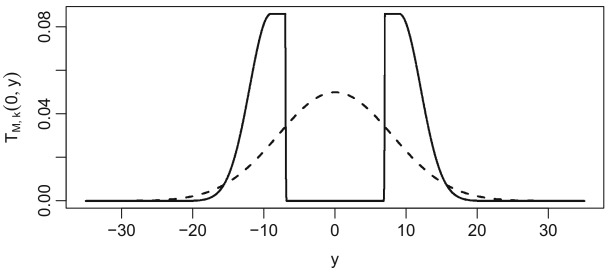


Fig. 4 Outermost proposal density ($j = M = 5$) for the AP method (solid) and the AG2 method (dashed) upon initialisation of the MCMC algorithms

illustrate how the novel Plateau proposals adapt to a given target distribution, we investigate properties of the resulting Markov chain during the initial iterations. For this investigation, two complementary scenarios are considered. First, we will study the resulting chain's coverage probability of a given target distribution's confidence region. Second, we will assess the first hitting time distribution for a chain to enter a high-probability region of the target distribution when initiated from a low-probability region.

We will compare the adaptive Plateau proposal algorithm with the adaptive Gaussian MCMC algorithm [27] using multivariate Normal target distributions. As the target distribution is Normal, performance should favour the adaptive Gaussian MCMC method due to similarity between the target and proposal distribution shape. We stress that a particular adaptation strategy is proposal-dependent, that is, here we compare the combined performance of the proposal *and* the adaptation procedure. We shall denote the adaptive Plateau proposal MCMC method as AP and the adaptive Gaussian proposal MCMC method proposed in [27] as AG2 (a variant of AG2, denoted as AG1, will be introduced later in Sect. 6). Following [27], the Gaussian proposal distributions in AG2 are adapted as follows: If the proposal with the largest standard deviation is under(over)-selected, then it is halved (doubled). Conversely, if the proposal with the smallest standard deviation is under(over)-selected, then it is doubled (halved). After either of these two updates, the other standard deviations are adjusted to be equidistant on a log-scale (base 2). For further algorithmic details on this adaptation scheme; see [27]. The adaptation interval for both methods was set to $L = 50$ and $P_n = -1$ (in Step 4, Algorithm 2) to trigger adaptations every 50 iterations. These settings were selected to make a fair comparison between the two MCMC methods. No burn-in was used in these simulations, as we are investigating the performance of the MCMC algorithms during the initial iterations.

The MCMC algorithms in these simulations were run with $M = 5$ trials. The proposals' standard deviations in the AG2 method are initialised with 2^{j-2} for proposal $j = 1, \ldots, M$. We reiterate that the adaptively chosen Plateau parameters are initialised with $\delta = \delta_1 = 1$, while σ and ς are fixed at 0.05 and 3 respectively. Moreover, we use the thresholds $\eta_1 = \eta_2 = 0.4$ for the AP method. The outermost proposal for both the AP and AG2 methods of this particular initialisation are illustrated in Fig. 4. Notice that the Gaussian proposal has slightly heavier tails that may give the AG2 an advantage by allowing larger moves compared to the AP method.

5.1 Scaling of the proposals and coverage probability

As mentioned above, we first consider the MCMC methods' coverage probabilities by monitoring how frequently a high-probability region is visited. We begin with a target distribution, ϖ_1, given by a 5-dimensional Normal distribution with mean zero and covariance matrix $\Sigma = \text{diag}(0.001, 0.1, 1, 10, 100)$. Note that the different magnitudes of the variances of the target should be captured directly by both MCMC methods. This is because the component-wise updates of the algorithms are along the same components of the target distribution; namely, on $x_1, x_2, \dots, x_5$. The number of independent repetitions of the MCMC simulations was set to $R = 5,000$ and the number of MCMC iterations performed for each run was $N = 10,000$. For each MCMC method, denote the chain for the rth repetition as $\boldsymbol{X}_0^{(r)}, \boldsymbol{X}_1^{(r)}, \dots, \dots, \boldsymbol{X}_N^{(r)}$ where $\boldsymbol{X}_j^{(r)} = \left(X_{j,1}^{(r)}, \dots, X_{j,5}^{(r)}\right)^T$ for $r = 0, \dots, R$. Further, denote the component-wise variances of the target ϖ_1 as $\sigma_1^2, \dots, \sigma_5^2$. Each repetition r of the chain was started at the origin $\boldsymbol{X}_0^{(r)} = \boldsymbol{X}_0 = (0, 0, 0, 0, 0)^T$ for both methods. For each repetition r, we report summaries of the running $(n = 1, \dots, N)$ empirical coverage probabilities of the component-wise $(k = 1, \dots, 5)$ confidence regions

$$C_n^{(r)}(k) := \frac{1}{n}\sum_{j=0}^{n} \mathbb{I}\left(\frac{(X_{j,k}^{(r)})^2}{\sigma_k^2} > z_1\right),$$

where z_1 is such that $\mathbb{P}(Z_1 > z_1) = 0.99$ and $Z_1 \sim \chi_1^2$ as well as the running empirical coverage probability of the joint 99% confidence region defined as

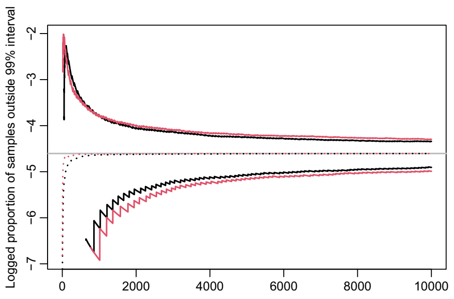
$$D_n^{(r)} := \frac{1}{n}\sum_{j=0}^{n} \mathbb{I}\left((\boldsymbol{X}_j^{(r)})^T \Sigma^{-1} \boldsymbol{X}_j^{(r)} > z_2\right),$$

where z_2 is such that $\mathbb{P}(Z_2 > z_2) = 0.99$ and $Z_2 \sim \chi_5^2$. Figure 5 presents the average as well as the 2.5% and 97.5% empirical quantiles of $\{C_n^{(r)}(k); r = 1, \dots, R\}$ for $k = 1, \dots, 5$ and of $\{D_n^{(r)}; r = 1, \dots, R\}$. Note that the vertical axes is on the log-scale to improve visibility of the results. In each reported summary, a point-wise interval is created by the 2.5% and 97.5% quantiles. The resulting empirical confidence intervals for the AP method are, for the majority of iterations, narrower than those for the AG2 method. This indicates that the AP method performs better than the AG2 method in terms of the coverage of the 99% marginal distributions of the target as well as the entire joint distribution. We reiterate that the AG2 method uses Gaussian component-wise proposals share exactly the same shape as the marginals of the target distribution. Despite this advantage, the AP method produces better results in terms of coverage.

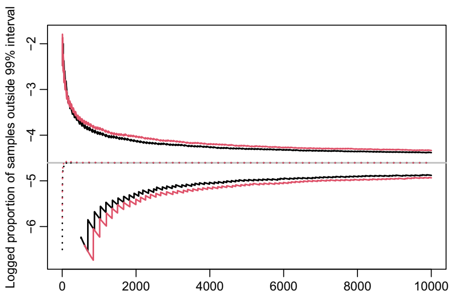
Next, we explore the performance of the adaptive MCMC algorithms for a target distribution, ϖ_2, with correlation across the components. The target distribution for this second example is taken to be a bivariate Normal distribution with mean zero and covariance matrix

$$\Sigma = \begin{pmatrix} 0.25 & 1.875 \\ 1.875 & 25 \end{pmatrix}.$$

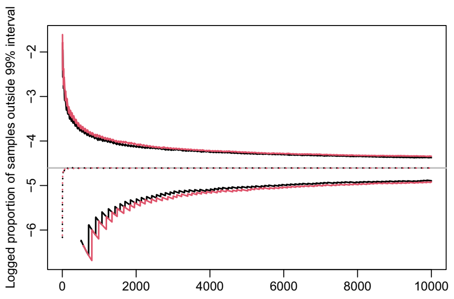
In this simulation, the component-wise MCMC proposal updates, for both the AP and AG2 methods, are not aligned with the principal components of the target distribution. Therefore, this target distribution should pose a greater difficulty to sample from in comparison to the previous target, ϖ_1. The results from this simulation study are presented in Fig. 6, where we report the analogous summaries of the running empirical coverage probabilities, as for the



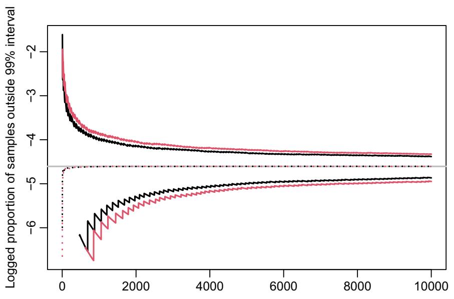
(a) Component $k = 1$



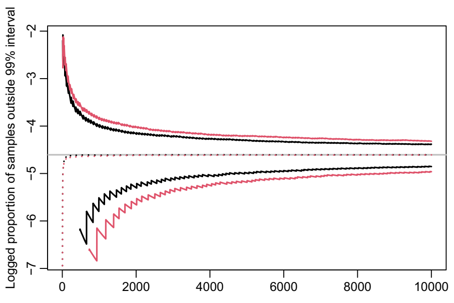
(b) Component $k = 2$



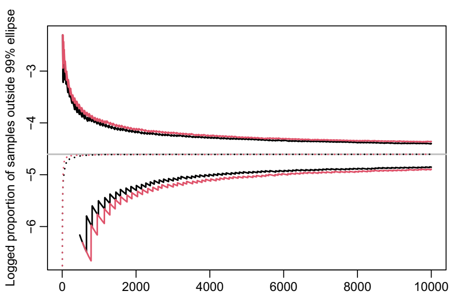
(c) Component $k = 3$



(d) Component $k = 4$



(e) Component $k = 5$



(f) Joint

Fig. 5 Uncorrelated 5-dimensional Gaussian Target, ϖ_1: **a**–**e** presents the average and 95% empirical quantiles of $\{C_n^{(r)}(k); r = 1, \dots, R\}$ for $k = 1, \dots, 5$ respectively. **f** present the average and 95% empirical quantiles of $\{D_n^{(r)}; r = 1, \dots, R\}$. Averages are represented by dotted lines and quantiles by solid lines. All results are presented on the log-scale. Black represents the AP method and red represents the AG2 method

previous example. Recall that both MCMC methods adapt independently per component. Therefore, the correlation between the targets' components is ignored in the proposals of the MCMC methods and their adaptations. Despite ignoring the correlation, both MCMC methods perform well in terms of coverage probabilities. As shown in Fig. 6, the AP method consistently outperforms the AG2 method, in terms of the width of the empirical confidence intervals.

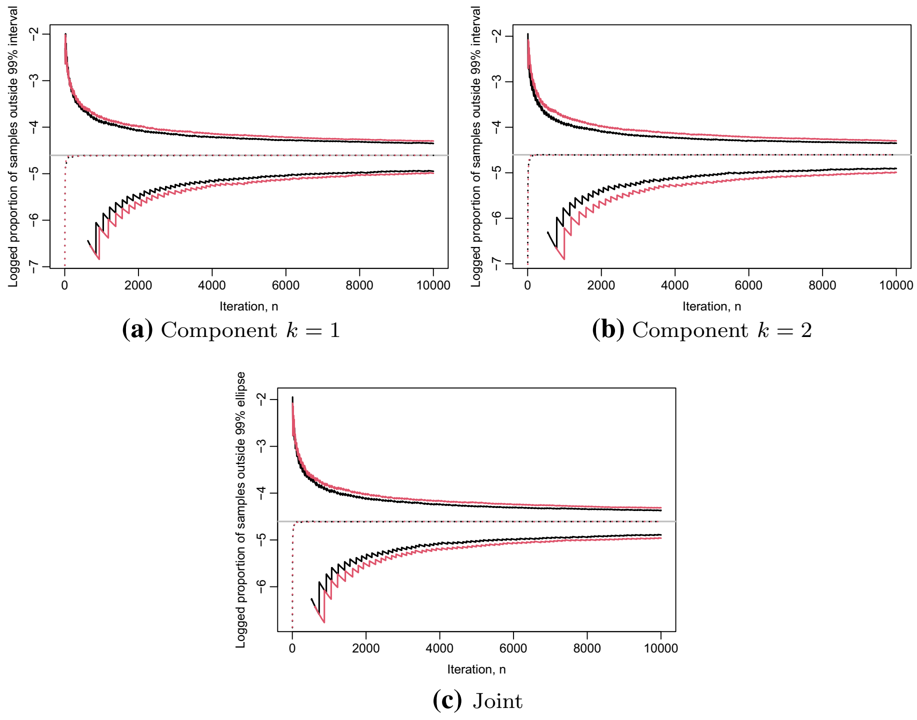


Fig. 6 Correlated 2-dimensional Gaussian Target, ϖ_2: **a** and **b** presents the average and 95% empirical quantiles of $\{C_n^{(r)}(k); r = 1, \ldots, R\}$ for $k = 1, 2$ respectively. **c** presents the average and 95% empirical quantiles of $\{D_n^{(r)}; r = 1, \ldots, R\}$. Averages are represented by dotted lines and quantiles by solid lines. All results are presented on the log-scale. Black represents the AP method and red represents the AG2 method

5.2 Moving to a high density region of target

We now investigate the adaptive MCMC methods' abilities to move toward areas of high target density, when initially started in a state outside that region. For this study, the target distribution is again taken as the correlated bivariate Normal distribution, ϖ_2. The number of independent repetitions of the MCMC runs was again set to $R = 5,000$ and the number of MCMC iterations performed each run was $N = 1,000$. Each run was initialised at the location very far away from the mean, $\boldsymbol{X}_0 = (50, 50)^T$.

For each repetition r of the MCMC method, the first time the Markov chain entered the (joint) 95% ellipse of the target is recorded as

$$J^{(r)} := \min\left(j \in \{0, \ldots, N\}; (\boldsymbol{X}_j^{(r)})^T \Sigma^{-1} \boldsymbol{X}_j^{(r)} < z_0\right),$$

where z_0 is such that $\mathbb{P}(Z_0 > z_0) = 0.95$ and $Z_0 \sim \chi_2^2$. The empirical distribution of the first hitting times $\{J^{(1)}, \ldots, J^{(R)}\}$ is summarised in the violin plots presented in Fig. 7 for AP and AG2 methods. Violin plots [13] are boxplots with kernel density estimates attached to the sides. We use violin plots as opposed to just boxplots to give a better illustration of the shape of the distributions. In the violin plots, the median is represented by a vertical line.

From Fig. 7 we observe that the AP method consistently moves into the 95% ellipse earlier than the AG2 method. Although the AG2 method does enter the ellipse earlier in

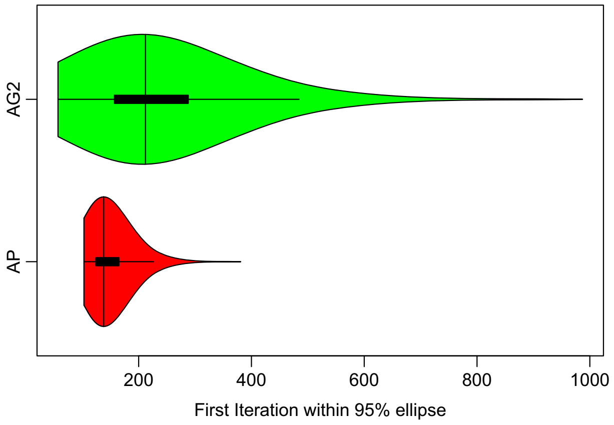


Fig. 7 Correlated 2-dimensional Gaussian Target, ϖ_2: Violin plot of first hitting time of the Markov chain into 95% ellipse. AP results represented in red and AG2 results in green (colour figure online)

some iterations, the AP method always (5, 000 out of 5, 000 iterations) entered the high-density region in less than 381 iterations. In comparison, in 517 out of 5, 000 iterations, the AG2 took more than 381 iterations to enter the ellipse.

In summary, the AP proposals adapt well to the scale of the target components during the initial steps of the algorithm. This was demonstrated in Sect. 5.1 using independent and correlated target components. Moreover, in Sect. 5.2 we showed that the AP method is able to effectively move into high-probability regions when initiated from low probability regions.

6 Results

In this section, we compare the long-term performance of the proposed adaptive component-wise, multiple-try MCMC method with other MCMC methods in simulations. The adaptive Plateau MCMC (AP) is compared to a Metropolis-Hastings algorithm with Gaussian proposals (MH) and two versions of an adaptive Gaussian MCMC (AG1 and AG2) as introduced in [27]. The difference between AG1 and AG2 is that AG1 uses (2) with $\alpha = 2.5$ (i.e., as in AP), while AG2 uses $\alpha = 2.9$ as is suggested in [27]. The proposal distributions in both AG1 and AG2 are adapted in the fashion outlined in Sect. 5.

For all simulations and methods with multiple trials, we fix the number of trials $M = 5$. Investigation of the method's performance for differing values of M is beyond the scope of this paper, which we leave for future work. However, interesting discussions in that direction already exist, see [19] for example.

The proposal standard deviation in the AG1 and AG2 methods are initialised at 2^{j-2} for $j = 1, \ldots, M$. The Plateau parameters values are initialised at $\delta = \delta_1 = 1$, $\sigma = 0.05$ and $\varsigma = 3$ with $\eta_1 = \eta_2 = 0.4$ for the AP method.

The proposal distributions used in the MH method depend on the particular target distribution and the choices used are summarised in Table 2. The various target distributions are now introduced.

6.1 Target distributions

In order to compare the aforementioned methods, we investigate their performances by applying them to sample from a variety of target distributions.

Mixture of Gaussians

Consider a mixture of two 4-dimensional Gaussians

$$\frac{1}{2}\mathrm{N}(\boldsymbol{\mu}_1, \Sigma_1) + \frac{1}{2}\mathrm{N}(\boldsymbol{\mu}_2, \Sigma_2),$$

where

$$\boldsymbol{\mu}_1 = (5, 5, 0, 0)^T, \quad \boldsymbol{\mu}_2 = (15, 15, 0, 0)^T$$

and

$$\Sigma_1 = \mathrm{diag}(6.25, 6.25, 6.25, 0.01), \quad \Sigma_2 = \mathrm{diag}(6.25, 6.25, 0.25, 0.01).$$

We refer to this target distribution as π_1.

Banana distribution

Consider the 8-dimensional “banana-shaped” distribution [10], which is defined as follows. Let f be the density of the 8D Normal distribution $\mathrm{N}(\mathbf{0}, \Sigma_3)$ with covariance given by $\Sigma_3 = \mathrm{diag}(100, 1, \ldots, 1)$. The density function of the banana distribution with non-linearity parameter $b > 0$ is given by $f_b = f \circ \phi_b$ where the function ϕ_b is

$$\phi_b(\boldsymbol{x}) = (x_1, x_2 + bx_1^2 - 100b, x_3, \ldots, x_8) \quad \text{for } \boldsymbol{x} \in \mathbb{R}^8.$$

The value of b determines the amount of non-linearity of ϕ_b. Here, we consider the target distribution $\pi_2 = f_{0.03}$, which leads to the unusual banana-shape of the first two components as shown in Fig. 8a.

Distributions perturbed by oscillations

Another target we consider is the perturbed 2-dimensional Gaussian, whose probability density function is given by

$$\pi_3(\boldsymbol{x}) \propto \exp\left[-\boldsymbol{x}^T A\boldsymbol{x} - \cos\left(\frac{x_1}{0.1}\right) - 0.5\cos\left(\frac{x_2}{0.1}\right)\right] =: \widetilde{\pi}_3(\boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in \mathbb{R}^2$$

where

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 3/2 \end{pmatrix}.$$

Figure 8c displays the un-normalised function $\widetilde{\pi}_3(\boldsymbol{x})$. Lastly, we also consider the following perturbed version of the 1D bi-stable distribution $x \mapsto Z^{-1}e^{-x^2+5x^2}$, whose PDF is given by

$$\pi_4(x) \propto \exp\left[-x^4 + 5x^2 - \cos\left(\frac{x}{0.02}\right)\right] \quad \text{for } x \in \mathbb{R}.$$

Figure 8d displays the PDF of $\pi_4(x)$ where the normalising constant is approximated by numerical integration.

Bayesian posterior distribution

A more complex target is given by the Bayesian posterior distribution originating from a localisation problem using multiple noisy sensor measurements in a wireless sensor network [18]. The problem involves using multiple observations from N_S sensor locations to estimate a target location $z \in \mathbb{R}^2$. Following [18], the true target location is $\boldsymbol{z}^* = (2.5, 2.5)^T$ and the $N_S = 6$ sensor locations are $\boldsymbol{h}_1 = (3, -8)^T$, $\boldsymbol{h}_2 = (8, 10)^T$, $\boldsymbol{h}_3 = (-4, -6)^T$, $\boldsymbol{h}_4 = (-8, 1)^T$, $\boldsymbol{h}_5 = (10, 0)^T$, and $\boldsymbol{h}_6 = (0, 10)^T$. The $N_O = 10$ observations at each sensor location are modelled as

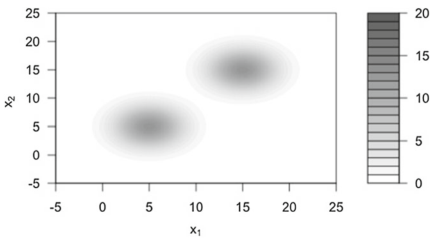
$$Y_{k,l} \sim \mathrm{N}\left(20 \log\left(\|z - \boldsymbol{h}_l\|_2\right), \zeta_l^2\right), \quad k = 1, \ldots, N_O \ ; \ l = 1, \ldots, N_S. \tag{4}$$

The observations $Y_{k,l}$ are assumed to be all independent of each other, and are drawn using the true location $z = z^*$ and standard deviations $\boldsymbol{\zeta}^* = (1, 2, 1, 0.5, 3, 0.2)^T \in \mathbb{R}^6$.

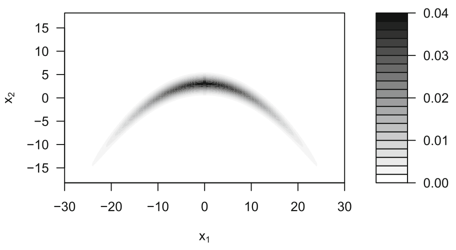
The objective is to estimate the target location z and the unknown sensor noise standard deviations $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_{N_S})$ from the given observations $Y = (Y_{k,l})_{\substack{k=1,\ldots N_O \\ l=1,\ldots N_S}}$. This is achieved using a Bayesian approach to compute the posterior distribution of z and $\boldsymbol{\zeta}$ given the observations Y. The prior for the target location, $z \in \mathbb{R}^2$, is uniform on $\mathcal{R}_z = (-30, 30)^2$ and similarly the prior for the standard deviations $\boldsymbol{\zeta}$ is uniform on $\mathcal{R}_\zeta = (0, 20)^{N_S}$. The MCMC target distribution is the posterior distribution of $\boldsymbol{x} := (z, \boldsymbol{\zeta})^T \in \mathbb{R}^{2+N_S}$ given the observations Y,

$$\pi_5(\boldsymbol{x}) := \pi(z, \boldsymbol{\zeta} | Y = y) \propto \prod_{k=1}^{N_O} \prod_{l=1}^{N_S} \frac{1}{\sqrt{2\pi\zeta_l^2}} \exp\left(-\frac{1}{2\zeta_l^2}\left(y_{k,l} - 20\log\left(\|z - \boldsymbol{h}_l\|_2\right)\right)^2\right)$$
$$\times \quad \mathbb{I}(z \in \mathcal{R}_z)\mathbb{I}(\boldsymbol{\zeta} \in \mathcal{R}_\zeta),$$

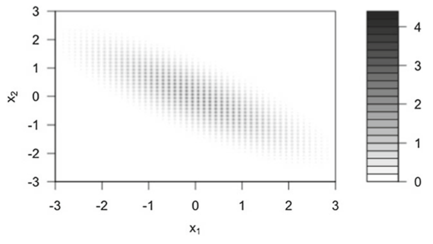
where $\mathbb{I}(\cdot)$ denotes the indicator function.



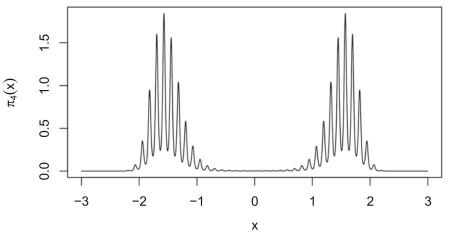
(a) Joint PDF of components 1 and 2 of π_1.



(b) Joint PDF of components 1 and 2 of π_2.



(c) Un-normalised PDF of π_3.



(d) PDF of π_4.

Fig. 8 Selected marginal density plots of simulation target distributions

6.2 Run parameters

Each simulation run of the MCMC methods was independently repeated $R = 200$ times and for each run a burn-in period of 50% of the MCMC iterations was used. During the burn-in period, the AP, AG1 and AG2 were allowed to adapt their proposals. For each repetition, all methods started at the same random initial position $\boldsymbol{x}_0$. The number of MCMC iterations, N, used for each method is presented in Table 1, which was determined by a trial run. In order to make fair comparisons, the number of MCMC iterations performed by Metropolis-Hastings algorithm is $d \times M$ times larger than the multiple-try versions. This is because multiple-try methods cycle over all d components and evaluate the target for M trials each iteration. This will ensure that the number of times the target distribution is evaluated by each MCMC method is the same and that the computational effort is approximately the same.

The proposal distribution in the MH method for each target is presented in Table 2. Note that these proposals are based on the target distribution, which would be typically be unknown in practice. The MH method should have an unfair advantage as its proposal is tuned to the target distribution. The particular scaling of $2.4/\sqrt{d}$ follows from [6]. The exception is the proposal distribution used for π_5, where we use the same proposal as used in [18].

6.3 Simulation results

For each target distribution, we compare the performance of the MCMC methods using measures based on the autocorrelations and jumping distances of the chain. We now define these measures mathematically. Denote the Markov chain produced by one of the MCMC methods for the rth independent repetition as $\boldsymbol{X}_0^{(r)}, \ldots, \boldsymbol{X}_N^{(r)}$ where $\boldsymbol{X}_i^{(r)} = (X_{i,1}^{(r)}, \ldots, X_{i,d}^{(r)})^T$ for $r = 1, \ldots, R$. Denote the component-wise variances of the target as $\sigma_1^2, \ldots, \sigma_d^2$.

Table 1 Number of MCMC iterations used in simulations for each target distribution

Target	Dimensions (d)	Adaptive	MH
π_1	4	4,000	80,000
π_2	8	10,000	400,000
π_3	2	3,000	30,000
π_4	1	3,000	15,000
π_5	8	10,000	400,000

Table 2 Proposal distributions used in the Metropolis-Hastings algorithm

Target	Proposal Distribution
π_1	$\frac{2.4}{\sqrt{4}}[0.5\mathrm{N}(\mathbf{0}, \Sigma_1) + 0.5\mathrm{N}(\mathbf{0}, \Sigma_2)]$
π_2	$\frac{2.4}{\sqrt{8}}\mathrm{N}(\mathbf{0}, \Sigma_3)$
π_3	$\frac{2.4}{\sqrt{2}}\mathrm{N}(\mathbf{0}, A^{-1})$
π_4	$2.4\mathrm{N}(0, 1)$
π_5	$\mathrm{N}(\mathbf{0}, I_8)$

We will use the integrated autocorrelation time (ACT) of the MCMC methods as a measure of performance. The ACT for the chain's kth component is given by

$$\mathrm{ACT}_k = 1 + \frac{2}{\sigma_k^2}\sum_{i=1}^{N} \mathrm{cov}(X_{0,k}, X_{i,k}) \ ,$$

provided that the chain is stationary so that $\boldsymbol{X}_0 \sim \pi$. For every repetition $r = 1, \ldots, R$ of the MCMC method, the integrated autocorrelation times are estimated based on the observed Markov chain $\boldsymbol{X}_0^{(r)}, \ldots, \boldsymbol{X}_N^{(r)}$ component-wise using the initial sequence estimator introduced in [7]. With slight abuse of notation, we will denote the resulting chain-based autocorrelation times by $\mathrm{ACT}_k^{(r)}$. Smaller autocorrelation times indicate that consecutive samples have lower correlation. Autocorrelation times are inversely proportional to the effective sample size [16, 17], which is another commonly used measure of performance. In fact, the effective sample size is often interpreted as the number of samples that would need to be (directly) drawn from the target in order to achieve the same variance as that from an estimator of interest using independent samples. Higher effective sample sizes and therefore lower autocorrelation times are desirable.

Another way of interpreting the ACTs is through the accuracy of a chain-based Monte Carlo integration. Moreover, the mean squared error of a Monte Carlo estimator can be expressed as a sum of the component-wise ACTs weighted by the component-wise variance. Consequently, a method that produces a chain with low ACTs is preferable. Further, the ACTs of an MCMC characterise the asymptotic variance (so-called time-averaged variance constant in this context) of a Monte Carlo estimator in the central limit theorem for Markov chains [1]. In fact, lower ACTs will lead to smaller time-averaged variance constants.

In practice, the target distribution is intractable and therefore the variances of the components are unknown. However, these variances are estimated by the initial sequence estimator method.

Another measure of performance is the chain's average squared jump distance (ASJD), which, for the kth component and repetition r, we define as

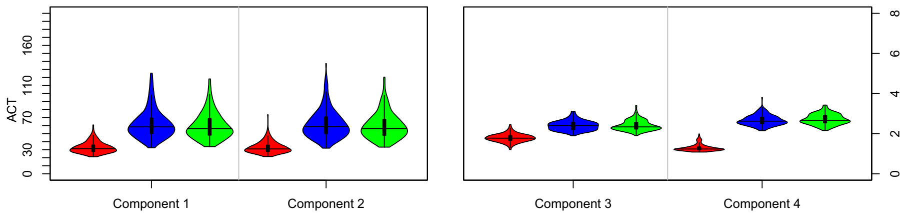
$$\mathrm{ASJD}_k^{(r)} = \frac{1}{N}\sum_{i=1}^{N} |X_{i,k}^{(r)} - X_{i-1,k}^{(r)}|^2.$$

The average squared jump distance measures the movement of the chain and also is linked with the acceptance rate of the MCMC method. Higher values of average squared jump distances are desired as it indicates larger moves and therefore more exploration of the space. We use the ASJD as a measure of the ability of an MCMC method to move around the state space.

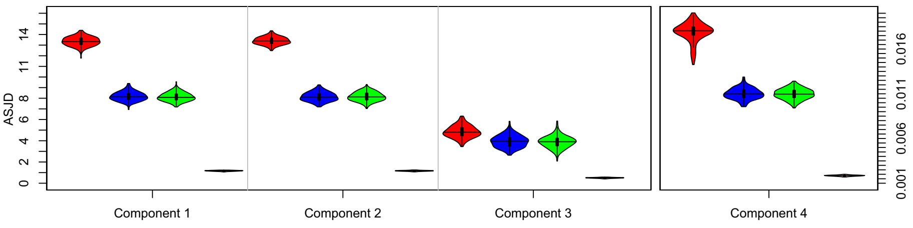
In summary, in the following results, we present the ACTs and the ASJD per component. The distribution of the ACTs and ASJDs over the repetitions, i.e., $\{\mathrm{ACT}_k^{(r)} : r = 1, \ldots, R\}$ and $\{\mathrm{ASJD}_k^{(r)} : r = 1, \ldots, R\}$, will be presented using violin plots.

6.3.1 Mixture of Gaussians

The results for the 4-dimensional mixture of two Gaussians target, π_1, are presented in Fig. 9. Fig. 9a indicates that the AP method achieves lower ACTs than the other methods for all components (including the MH method, which is not included in this figure due to very high ACTs). Further, the range of ACT values suggest that the AP method consistently produces MCMC chains with lower ACTs.



(a) Autocorrelation times



(b) Average square jump distance

Fig. 9 Mixture of Gaussians, π_1: Distribution of the ACTs and ASJDs of MCMC methods for target π_1. Red is AP, blue is AG1, green is AG2 and pink is MH (colour figure online)

In terms of the movement of the MCMC chains, the AP method outperforms the other methods for this target distribution. In fact, the ASJDs presented in Fig. 9b show that the AP method moving around the state space in larger jumps than the other methods. Since the AP and AG1 methods use the same weight function as discussed in Sect. 2.1 this advantage is due to using the Plateau proposals in contrast to Gaussian proposals. These results suggest that the AP method is able to move between the two Gaussians in the target density efficiently.

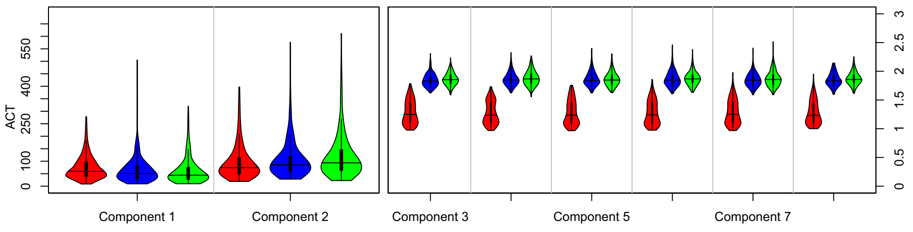
6.3.2 Banana distribution

The target, π_2, is a difficult distribution to sample from due to the wide-ranging variances across the components and the unusual banana-shape of the first two components. The ACTs for the methods, presented in Fig. 10a, show similar results across the AP, AG1 and AG2 for the first two components. However, for the remaining components, the AP is achieving notably smaller ACTs. The ACT results for the MH are substantially larger for all components, and thus are not included in this figure. As an indication, the median ACTs for components 1 to 8 respectively are: 1131.74, 2066.35, 54.24, 54.37, 54.34, 54.03, 54.74, 54.47 for the MH method to 2 decimal places.

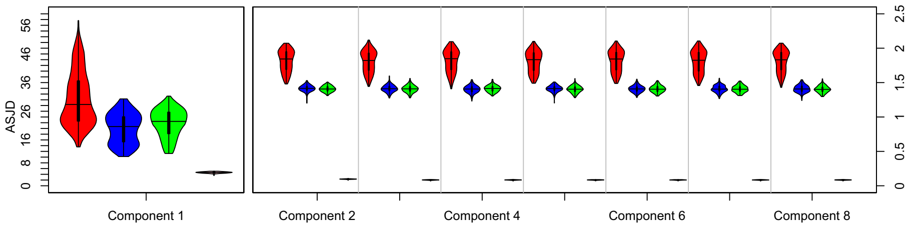
The ASJDs for the methods presented in Fig. 10b. The AP method again outperforms the other methods by achieving higher ASJDs for all components. Note that for the first component, the wide range of jumping distance produced when using the Plateau proposals. This suggests that the AP method is able to navigate the banana-shape in the first component easily.

6.3.3 2D perturbed distribution

For the perturbed 2-dimensional distribution, π_3, the perturbations represent local modes where MCMC methods may potentially get stuck. Again, the AP method's ability to move



(a) Autocorrelation times



(b) Average square jump distance

Fig. 10 Banana Distribution, π_2: Distribution of the ACTs and ASJDs of MCMC methods for target π_2. Red is AP, blue is AG1, green is AG2 and pink is MH (colour figure online)

slightly larger distances, as shown in Fig. 11b, gives it a slight advantage over the other methods. This ability to jump further may explain the lower ACTs for the AP method as depicted in Fig. 11a.

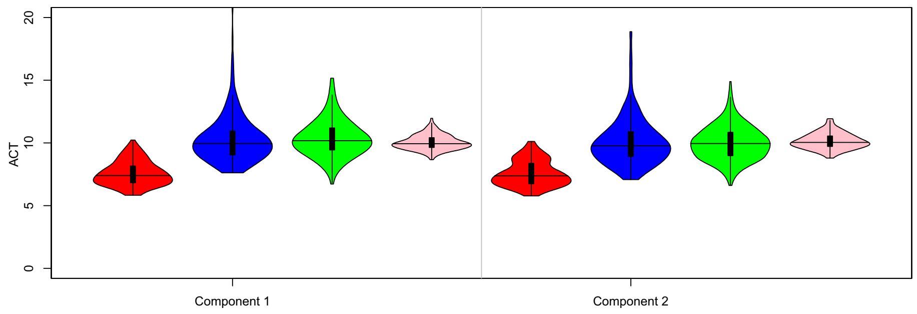
6.3.4 1D perturbed distribution

Similar to π_3, the oscillations in π_4 are areas where an MCMC may get stuck. The ACTs for the AP, AG1 and AG2 method are presented in Fig. 12a. The AP achieves the lowest ACTs; however, there are a few outliers which may indicate a few runs where the sampler got stuck in the local modes. This may also be the case for the AG1 method. For the MH method, the ACTs (not presented in the figure) are extremely large in comparison to the other methods – with a median of 178.54 and a range of 111.62 to 457.56 to 2 decimal places.

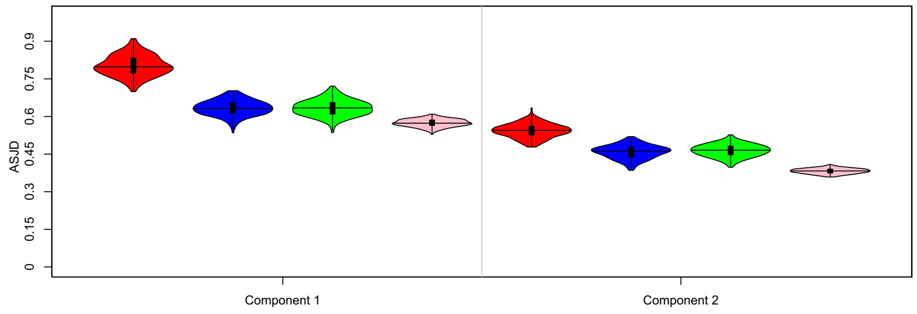
The ASJDs for the AP method, is on average jumping also twice the distance of the AG1 and AG2 methods – see Fig. 12b. Again, there are some outlying ASJDs for the AP method, which may indicate some repetitions where the MCMC got stuck in local modes.

6.3.5 Posterior distribution

The posterior distribution, π_5, poses a difficult distribution to sample from due to the non-linearity of the parameters. The ACTs for the MCMC methods (except MH) are presented in Fig. 13a. The AP method consistently achieves a lower ACT than the AG1 and AG2 method, especially for the variance parameters (components 3 to 8). In terms of ASJDs, all methods give similar performance – see Fig. 13b, with a slight advantage to using the AP method which achieves higher ASJD albeit infrequently. The ACT and ASJD results for the MH method are not presented in Fig. 13a. The median ACT for components 1 to 8 range from 19730 to 37430. Further, the median ASJDs range from 2.9×10^{-8} to 4.4×10^{-6} over

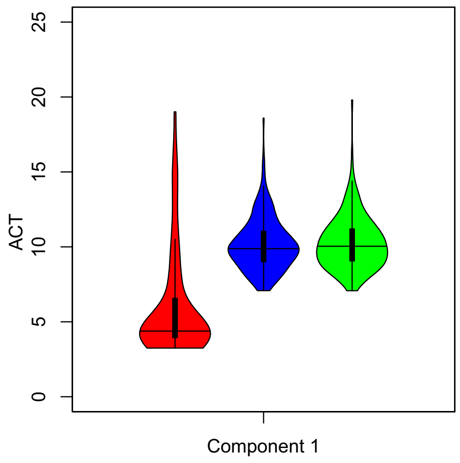


(a) Autocorrelation times

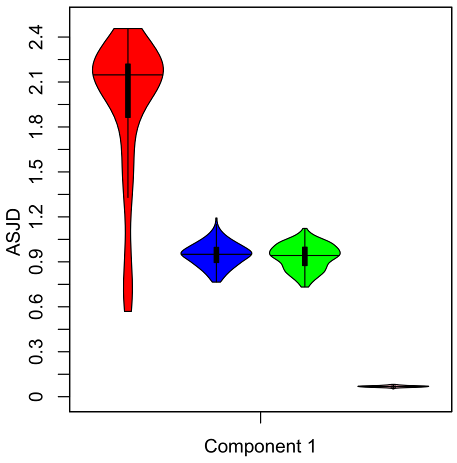


(b) Average square jump distance

Fig. 11 2D Perturbed Distribution, π_3: Distribution of the ACTs and ASJDs of MCMC methods for target π_3. Red is AP, blue is AG1, green is AG2 and pink is MH (colour figure online)

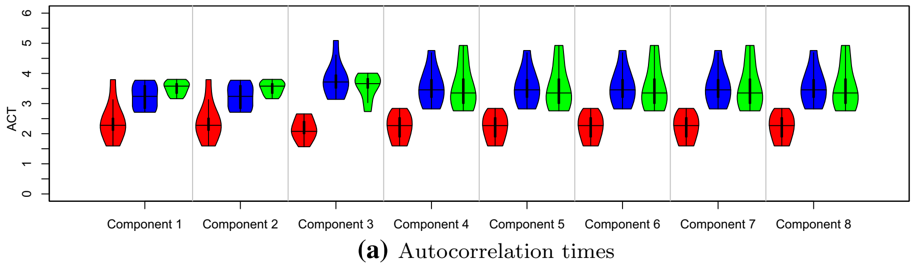


(a) Autocorrelation times



(b) Average square jump distance

Fig. 12 1D Perturbed Distribution, π_4: Distribution of the ACTs and ASJDs of MCMC methods for target π_4. Red is AP, blue is AG1, green is AG2 and pink is MH (colour figure online)



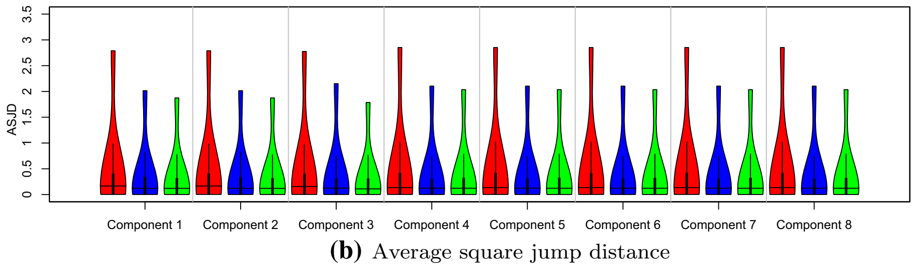


Fig. 13 Posterior Distribution, π_5: Distribution of the ACTs and ASJDs of MCMC methods for target π_5. Red is AP, blue is AG1, green is AG2

all 8 components. Both the high ACTs and very low ASJDs indicate that the MH method performed poorly.

6.4 Comparing computational speed

In this section, we compare the computational effort of the AP and AG methods. For this comparison, we use the 8-dimensional banana target distribution, π_2 and investigate the time to run the MCMC methods with a different number of trials M and MCMC steps N. Specially, we run a single simulation with a unique combination of $M \in \{1, 4, 5, 10\}$ and $N \in \{5000, 10000, 50000, 100000\}$. Each simulation is repeated 1, 000 times. We compare the AP and AG methods with $\alpha = 2.9$ both without and with the adaptation. The methods are written in C++.

The results presented in Tables 3 and 4 reports the average effort to run each MCMC methods over 1, 000 iterations. The reported effort is relative to the effort to generate the MCMC samples from the AG method.

Table 3 shows that the AP and AG methods, with and without adapting, are all comparable in terms of effort for all values of N. Table 4 suggests that keeping N fixed and increasing the number of trial M does not lead to any one method being more efficient than the others. We

Table 3 Computational effort of MCMC algorithms using $M = 5$ trials divided by the effort using the AG method

	$N = 5,000$	10,000	50,000	100,000
AP	1.0092	1.0208	1.0118	1.0064
AP (no adapt)	1.0079	1.0168	1.0069	1.0051
AG	1.0000	1.0000	1.0000	1.0000
AG (no adapt)	0.9975	1.0043	1.0038	1.0048

Table 4 Computational effort of MCMC algorithms using $N = 1000$ MCMC steps divided by the effort using the AG method

	$M = 1$	3	5	10
AP	1.1197	1.0320	1.0208	0.9939
AP (no adapt)	1.1228	1.0343	1.0168	0.9909
AG	1.0000	1.0000	1.0000	1.0000
AG (no adapt)	1.0421	0.9990	1.0043	1.0005

conclude that both the AP and AG methods are comparable in terms of computation time, as there are negligible differences between the two methods.

7 Generalisation of plateau proposals

In this section we explain why the family of Plateau proposals are suitable within multiple-try MCMC methods. As mentioned in Sect. 3, the Plateau proposals are designed to be non-overlapping and simultaneously have no (or small) gaps between the individual proposals. The non-overlapping feature ensures that trial samples are drawn from separate areas meaning that the chain explores the area efficiently. The feature of having no gaps ensures that there are no regions between proposals that will not be explored – meaning that the entire combined support of the proposals is being explored with a non-negligible probability.

We now introduce measures of overlap and gap between two trial distributions. In this section, we simplify the setting to two contiguous trial distributions rather than using M distributions, to make clear the overlap and gap measures. The intuition carries over to the full case with M trial distributions. The overlap and gap between distributions is illustrated in (i) the Gaussian case and (ii) the Plateau case which are now defined. Let A and B two independent random variables. In the Gaussian case let $A \sim \mathrm{N}(0, s^2)$ and $B \sim \mathrm{N}(\Delta, s^2)$ for some $\Delta > 0$. In the Plateau case let $A \sim f(\cdot, \mu = 0, \delta = \Delta/2, \sigma_1 = \sigma, \sigma_2 = \sigma)$ and $B \sim f(\cdot, \mu = \Delta, \delta = \Delta/2, \sigma_1 = \sigma, \sigma_2 = \sigma)$ where the PDF f is defined by Eq. (3). Recall that in the PDF f, the parameter μ is the location of the centre of a single plateau and σ_1 (σ_2) determines the rate of exponential decay to the left (right) side of the plateau. As suggested earlier, and used in all numerical simulation conducted in this paper, we set $\sigma_1 = \sigma_2 = \sigma$. The Gaussian and Plateau cases are illustrated in Fig.14.

We measure the overlap between the distributions of A and B as

$$p_{\text{overlap}} := \mathbb{P}(B < m_p) + \mathbb{P}(A > m_p),$$

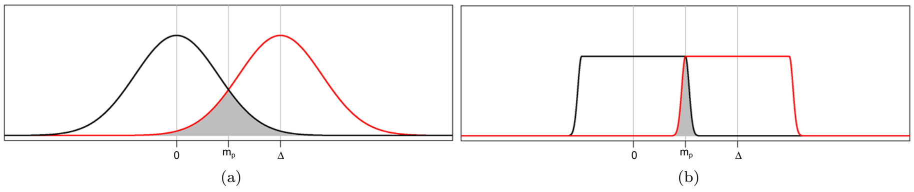


Fig. 14 PDF illustrations for random variables A (black) and B (red) for **a** the Gaussian case and **b** the Plateau case. Shaded grey area represents the overlap measure, p_{overlap} (colour figure online)

where $m_p = 0.5(\mu_1 + \mu_2)$, where μ_1 and μ_2 denote the mean values of A and B, respectively. The overlap measure is illustrated in Fig.14 by the shaded grey areas for both the Gaussian and Plateau case. Low values of p_{overlap} are ideal.

Next, we define the gap between the distributions A and B as follows: Define the interval $\mathfrak{G} = (m_p - \psi, m_p + \psi)$ for some $\psi > 0$ that satisfies

$$\mathbb{P}\left(\{A \notin \mathfrak{G}\} \cap \{B \notin \mathfrak{G}\}\right) = 0.9. \tag{5}$$

We define the gap distance as ψ – the half-width of the interval $\mathfrak{G}$. Equation (5) can be read as: *the probability of neither A or B being sampled from* $(m_p - \psi, m_p + \psi)$ *is high*. Therefore, it is ideal to have $\psi > 0$ as small as possible, so that there is a very narrow interval between the distributions A and B where neither is likely to be sampled from. Note that the probability value of 0.9 in Eq. (5) is ad-hoc. Other high values could be used, however, the intuitive results of this section remain the same.

The Plateau distributions are initialised with parameters $\delta = 1$ and $\sigma = 0.05$. Using these parameters gives the values $p_{\text{overlap}} = 0.028$ and $\psi = 0.059$. To compare with these values, the gap distances and overlap measures are computed in the Gaussian case for $\Delta \in (0, 5)$ and $s \in (0, 2.5)$. The overlap measures and gap distances are presented as heatmaps in Fig.15. Contour lines representing p_{overlap} and ψ for the initialised Plateau proposals are included on the heatmaps for comparison.

It is possible to select Δ and s in the Gaussian case to simultaneously obtain the same overlap and gap measures achieved in the initial Plateau case. The values of Δ and s that satisfies $p_{\text{overlap}} = 0.028$ and $\psi = 0.059$ are: $\Delta = 0.286$ and $s = 0.0757$. Figure 16 presents the Gaussian case with $\Delta = 0.286$ and $s = 0.0757$ and the Plateau case with $\delta = 1$ and $\sigma = 0.05$ which achieve the same overlap and gap measure. The mid-point of the distributions means has been centred at zero. From Fig. 16 it is clear that sampling region of the Plateau is far broader than the narrow Gaussians. As a crude numerical illustration, consider the range of the union of the symmetric 95% density regions of A and B. In the Gaussian case this range $(-0.292, 0.292)$ in comparison to the Plateau case $(-2.010, 2.010)$. This indicates that the Plateaus have a far greater coverage, and thus offer the capability to explore a larger region than using Gaussians. In order to achieve the same sampling coverage area as the two Plateaus, more Gaussians would need to be used. Increasing the number of distributions, M, comes at a computational cost as more proposals need to drawn and trial weights calculated which requires computation of the target – see steps 4 and 5 in Algorithm 1.

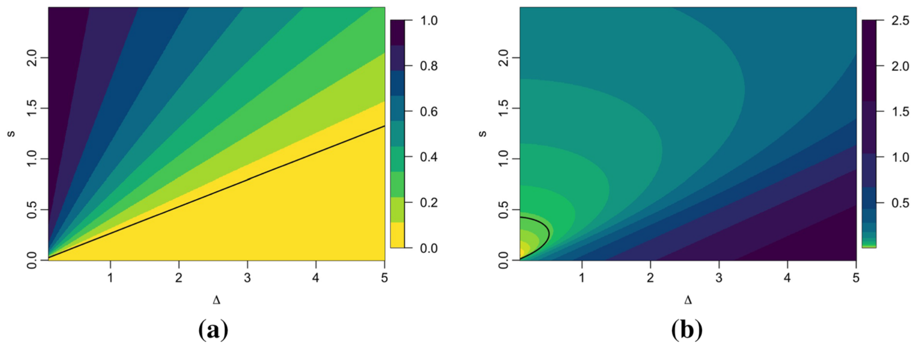


Fig. 15 Heatmaps of **a** the overlap measure and **b** gap distance in the Gaussian case. The black contour lines indicate the **a** $p_{\text{overlap}} = 0.028$ and **b** $\psi = 0.059$, obtained in the Plateau case with $\delta = 1$ and $\sigma = 0.05$

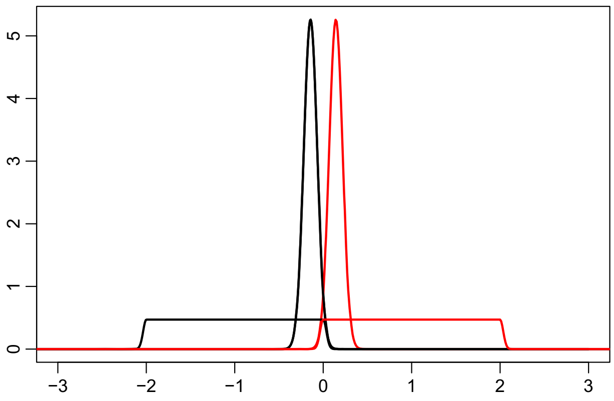


Fig. 16 PDFs of **a** the Gaussian case with $\Delta = 0.286$ and $s = 0.0757$ and **b** the Plateau case with $\delta = 1$ and $\sigma = 0.05$. The mid-point of the distribution means is set equal to zero. Both cases achieve overlap $p_{\text{overlap}} = 0.028$ and $\psi = 0.059$

In summary, using Plateau shaped proposals in multiple-try MCMC methods, efficiently explore the space by not overlapping and by not having large gaps between contiguous trial distributions. Further, the combined wide support of Plateaus means that few trial distributions (M) are required to sample a large region.

The overlap measure and gap distance used in this section were constructed specifically to compare the Gaussian and Plateau proposal shapes. Comparing the Plateau distributions against non-Gaussian distributions may require an extension to our definitions or creation of new ones. A more in-depth study as to how the overlap and gap properties of proposals effect MCMC results is outside the scope of this paper and is left to future work.

8 Conclusion

In this paper, we have introduced Plateau distributions as a novel-class of proposal distributions for the use with component-wise, multiple-try Metropolis MCMC. These proposal distributions are a combination of uniform distributions, leading to a family of distributions with non-overlapping supports. The notion of using non-overlapping proposals in multiple-try MCMC methods is intuitive and, in fact, motivated as means to counter the disadvantages (e.g., inefficient proposing of trials) of severely overlapping proposal distributions such as Gaussians. Moreover, the class of Plateau distributions are simple to implement for use as proposals in MCMC methods and are straightforwardly combined with the simple, yet highly effective, adaptation procedure presented in Sect. 4. As mentioned in the introduction, the novelty of this work lies in both the Plateau proposals and the bespoke adaptation method. The designed adaptation method takes advantage of the non-overlapping proposals to better explore the space and "scale" the proposals to the target distribution. The advantages of our proposed algorithm over Gaussian proposals with a similar adaptation method was presented in simulations in Sect. 5. Further, in Sect. 6.4 we showed that the computational effort of using Plateau proposals is comparable to using Gaussians.

We have demonstrated that using the Plateau proposal distributions with the suggested adaptation leads to MCMC methods that perform well for a variety of target distributions. In fact, the results indicate that using our method produces MCMC chains that explore the state space better with lower autocorrelation times, when compared to other adaptive multiple-try methods with greatly overlapping Gaussian proposals. Furthermore, the simulation results suggest that the Plateau proposals are able to efficiently sample from target distributions with distance modes, complex shapes, and many nearby modes.

The results and the simplicity of their design make the Plateau proposals appealing for general use in component-wise, multiple-try MCMC algorithms. As a matter of fact, the introduced class of Plateau distributions is one type of non-overlapping proposals. Further research may investigate other types of non-overlapping proposals, which may have multiple interacting trials (e.g., see [3]) and may be asymmetric. Further theoretical research is required to determine the mixing properties of the MCMC chain produced by these Plateau proposals and adaptation procedure.

In the simulations results presented in Sect. 6, using $M = 5$ trials worked well in all examples. Further work is required to investigate the MCMC performance as both the number of trials and the target dimension changes. This investigation goes beyond studying only Plateau proposals, and therefore is left to future work. Some theoretical results for multiple-try methods concerning the limiting behaviour $M \to \infty$, are available in [15].

Acknowledgements We are grateful for the insightful and useful comments provided by the anonymous referees that greatly improved the quality of our paper.

Funding Open Access funding enabled and organized by Projekt DEAL.

A A note of convergence of adaptive component-wise multiple-try algorithms

The convergence (in total variation distance) of algorithms of the form of Algorithm 1 described in Sect. 4 has been proven in [27]. The proof of convergence is ensured by the algorithm, both the MCMC algorithm and the adaptation procedure, satisfying two conditions: diminishing adaptation and containment. As mentioned earlier, diminishing adaptation is satisfied by adapting with probability $P_n = \max\left\{0.99^{n-1}, 1/\sqrt{n}\right\}$. For containment to hold two technical, but not practical, modifications are required – these follow directly from [27] and are presented as quotations below with altered notation. The first modification is to

> "... choose a very large nonempty compact set $K \subset \mathcal{X}$ and force $\boldsymbol{X}_n \in K$ for all n. Specifically, we reject all proposals $\boldsymbol{Y} \notin K$ (but if $\boldsymbol{Y} \in K$, then we still accept/reject $\boldsymbol{Y}$ by the usual rule)..."

The second modification which is altered for our proposed Plateau distributions is

> "... choose a very large constant Δ and a very small constant $\epsilon > 0$ and force the proposal width δ to always be in $[\epsilon, \Delta]$..."

The proof then follows Sect. 3.5 in [27].

References

1. Asmussen, S., Glynn, P. W.: Stochastic Simulation: Algorithms and Analysis. Springer, (2007)
2. Brooks, S., Gelman, A., Jones, G.L., Meng, X.-L. (eds.): Handbook of Markov chain Monte Carlo. CRC Press, Boca Raton (2011)
3. Casarin, R., Craiu, R., Leisen, F.: Interacting multiple try algorithms with different proposal distributions. Stat. Comput. **23**(2), 185–200 (2013)
4. Casella, G., George, E.I.: Explaining the gibbs sampler. Am. Stat. **46**(3), 167–174 (1992)
5. Craiu, R.V., Lemieux, C.: Acceleration of the Multiple-Try Metropolis algorithm using antithetic and stratified sampling. Stat. Comput. **17**(2), 109–120 (2007)
6. Gelman, A., Roberts, G., Gilks, W.: Efficient Metropolis jumping rules. Bayesian Stat. **5**(599–608), 42 (1996)
7. Geyer, C.J.: Practical Markov chain Monte Carlo. Stat. Sci. **7**(4), 473–483 (1992)
8. Gilks, W. R., Richardson, S., Spiegelhalter, D.: Markov chain Monte Carlo in practice. Chapman and Hall/CRC, (1995)
9. Giordani, P., Kohn, R.: Adaptive independent Metropolis-Hastings by fast estimation of mixtures of normals. J. Comput. Graph. Stat. **19**(2), 243–259 (2010)
10. Haario, H., Saksman, E., Tamminen, J.: Adaptive proposal distribution for random walk Metropolis algorithm. Comput. Stat. **14**(3), 375–395 (1999)
11. Haario, H., Saksman, E., Tamminen, J.: An adaptive Metropolis algorithm. Bernoulli **7**(2), 223–242 (2001)
12. Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57**(1), 97–109 (1970)
13. Hintze, J.L., Nelson, R.D.: Violin plots: A box plot-density trace synergism. Am. Stat. **52**(2), 181–184 (1998)
14. Johnson, A.A., Jones, G.L., Neath, R.C.: Component-wise Markov chain Monte Carlo: uniform and geometric ergodicity under mixing and composition. Stat. Sci. **28**(3), 360–375 (2013)
15. Jun, W.H.W., Liu, S., Liang, F.: The multiple-try method and local optimization in Metropolis sampling. J. Am. Stat. Assoc. **95**(449), 121–134 (2000)
16. Kong, A.: A note on importance sampling using standardized weights. University of Chicago, Dept. of Statistics, Technical Report, 348, (1992)
17. Kong, A., Liu, J.S., Wong, W.H.: Sequential imputations and Bayesian missing data problems. J. Am. Stat. Assoc. **89**(425), 278–288 (1994)
18. Martino, L.: A review of multiple try mcmc algorithms for signal processing. Dig. Signal Process. **75**, 134–152 (2018)
19. Martino, L., Louzada, F.: Issues in the multiple try metropolis mixing. Comput. Stat. **32**(1), 239–252 (2017)
20. Martino, L., Read, J.: On the flexibility of the design of multiple try Metropolis schemes. Comput. Stat. **28**(6), 2797–2823 (2013)
21. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. J. Chem. Phys. **21**(6), 1087–1092 (1953)
22. Neal, R. M. et al.: Mcmc using hamiltonian dynamics. In Brooks, S., Gelman, A., Jones, G., Meng, X.-L. editors, Handbook of Markov chain Monte Carlo. CRC press, (2011)
23. Roberts, G.O., Rosenthal, J.S.: Optimal scaling for various Metropolis-Hastings algorithms. Stat. Sci. **16**(4), 351–367 (2001). (**11**)
24. Roberts, G.O., Rosenthal, J.S.: Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. J. Appl. Probab. **44**(2), 458–475 (2007)
25. Roberts, G.O., Tweedie, R.L.: Exponential convergence of Langevin distributions and their discrete approximations. Bernoulli **2**(4), 341–363 (1996)
26. Rosenthal, J. S.: Optimal proposal distributions and adaptive MCMC. In Brooks, S., Gelman, A., Jones, G., Meng, X.-L. editors, Handbook of Markov chain Monte Carlo. CRC press, (2011)
27. Yang, J., Levi, E., Craiu, R.V., Rosenthal, J.S.: Adaptive component-wise multiple-try Metropolis sampling. J. Comput. Graph. Stat. **28**(2), 276–289 (2019)