Aust. N. Z. J. Stat. 2020 doi: 10.1111/j.1467-842X.XXX

Accelerating adaptation in the adaptive Metropolis-Hastings random walk algorithm

Simon E.F. Spencer[1]

University of Warwick

Summary

The Metropolis-Hastings random walk algorithm remains popular with practitioners due to the wide variety of situations in which it can be successfully applied and the extreme ease with which it can be implemented. Adaptive versions of the algorithm use information from the early iterations of the Markov chain to improve the efficiency of the proposal. The aim of this paper is to reduce the number of iterations needed to adapt the proposal to the target, which is particularly important when the likelihood is time-consuming to evaluate. First, the accelerated shaping algorithm is a generalisation of both the adaptive proposal and adaptive Metropolis algorithms. It is designed to remove misleading information from the start of the chain from the estimate of the covariance matrix of the target. Second, the accelerated scaling algorithm rapidly changes the scale of the proposal to achieve a target acceptance rate. The usefulness of these approaches is illustrated with a range of examples.

Key words: MCMC; adaptive Metropolis; adaptive proposal; shaping; scaling; banana

1. Introduction

The Metropolis-Hastings random walk (MHRW) algorithm (Metropolis et al. 1953; Hastings 1970) is a Markov chain Monte Carlo (MCMC) algorithm that has an enduring popularity with practitioners due to the ease of implementation and the wide variety of circumstances in which it is applicable. Adaptive versions of the algorithm (Haario, Saksman & Tamminen 1999; Haario et al. 2001) automatically tune the proposal covariance matrix to improve the mixing. Landmark papers have shown that, for a d dimensional target with covariance matrix $\mathbf{\Sigma}$, in a range of circumstances the optimal proposal covariance matrix is $\frac{2.38^2}{d}\mathbf{\Sigma}$, leading to an optimal acceptance rate of 0.234 (Gelman, Roberts & Gilks 1996; Roberts, Gelman & Gilks 1997; Roberts & Rosenthal 2001). Curiously this acceptance rate has also been proved to be optimal for some other MCMC proposals Lee et al. (2018). One

[1] Department of Statistics, University of Warwick, Coventry, CV4 7AL, UK
Email: s.e.f.spencer@warwick.ac.uk

Acknowledgment. The author gratefully acknowledges funding by MRC grant MR/P026400/1, EPSRC grant EP/R018561/1 and funding of the NTD Modeling Consortium by the Bill and Melinda Gates Foundation [OPP1156227, OPP1186851, OPP1184344]. He also wishes to thank Massey University in New Zealand for hosting him during 2018/2019 and Lloyd Chapman for providing comments on the draft manuscript.

Prepared using anzsauth.cls *[Version: 2018/01/30 Version 9]*

drawback of the adaptive MHRW algorithm is that it can take a large number of iterations to adapt to the target. Accelerating the speed at which adaptations occur during the burn-in phase of the MHRW algorithm is the subject of this paper.

Another drawback of adaptive algorithms is that the process of learning the shape of the target from the history of the chain destroys the Markov property and therefore it can be challenging to prove that an adaptive algorithm converges to the correct target. Indeed, in some examples adaptive algorithms have been shown to converge towards an incorrect distribution (Haario, Saksman & Tamminen 1999; Roberts & Rosenthal 2007; Atchadé & Rosenthal 2005). Many adaptive algorithms in the literature come with accompanying proofs of asymptotic convergence, and may contain features designed to facilitate these proofs rather than rapid adaptation, leaving scope for improvement in most practical applications. One existing approach avoiding such difficulties is Yang & Rosenthal (2017) where the adaptation is stopped once the mixing appears close to optimal. The authors step-by-step approach identifies a first adaptive phase, a transient phase (for travelling to the mode), a second adaptive phase and finally a sampling phase with no adaptation, which guarantees convergence to the correct target. Although this approach provides great scope for accelerating adaptation, the focus of the paper is on diagnostics to identify the phases and they do not explore accelerating the rate of adaptation in detail.

Better mixing MCMC algorithms than the MHRW exist (see Robert et al. 2018, for a review), but generally require more effort to implement. Examples include gradient-based approaches, e.g. Hamiltonian Monte Carlo (Neal 2011), Metropolis-adjusted Langevin algorithm (Roberts & Tweedie 1996), etc. and the delayed rejection adaptive Metropolis algorithm (Haario et al. 2006). Furthermore, the burn-in phase of an MCMC algorithm can be greatly reduced by starting the chain from the posterior mode. However there are circumstances in which it is very time-consuming or unhelpful to calculate gradients of the target. For example, when there are discrete parameters, large amounts of missing data are being imputed, the likelihood function takes a long time to evaluate, or there are discontinuities in the target. Any one of these circumstances mean that both numerical optimisation to find the posterior mode and gradient-based MCMC algorithms becoming challenging to implement and practitioners frequently revert to the MHRW algorithm. An important motivating application is the problem of fitting non-linear systems of differential equations to time series data, such as when modelling the spread of infectious diseases (see for example Keeling & Rohani 2011; Hollingsworth et al. 2015). The model equations can only be solved numerically and so posterior gradients are not available; the posterior mode is challenging to obtain and the likelihood is time-consuming to evaluate and so a large amount of computation time is required to fit the model.

This paper focusses on methods to shorten the burn-in phase of the adaptive MHRW algorithm by making the adaptation occur more rapidly. It outlines a general algorithm for shaping the MHRW proposal, which includes both the Adaptive Proposal (Haario, Saksman & Tamminen 1999) and the Adaptive Metropolis (Haario et al. 2001) algorithms as special cases. Furthermore, in Section 2.3, a rescaling approach is discussed that uses the Robbins-Munro algorithm to achieve a target acceptance rate. As the number of MCMC samples increases, the algorithms described in this paper converge towards existing adaptive algorithms that approach optimal mixing, but as illustrated by the examples in Section 3, they approach the optimal proposal more rapidly. The adjustments that are proposed are straight-forward to implement without substantial extra coding. The number of user-specified parameters has been kept to a minimum so that the resulting algorithm is not too problem specific and can be applied to a wide range of well-behaved problems with little tuning. However there are many examples of problems for which the MHRW algorithm is not a good choice (for example when the target has heavy tails or there are multiple separated local modes) and for these problems the proposed algorithm will not generate good mixing, and may even fail to converge. Alternative algorithms that explore mutli-modal targets should be used instead, such as simulated tempering (Geyer 1991; Tawn, Roberts & Rosenthal 2020) or parallel tempering (Marinari & Parisi 1992; Miasojedow, Moulines & Vihola 2013; Tawn & Roberts 2019). For a discussion of these and related methods, see for example Tawn (2017).

2. Methods

2.1. Accelerated shaping

In a landmark paper, Haario et al. (2001) developed a Gaussian random walk proposal and proved that the resulting MCMC approaches the correct target. This algorithm is termed the Adaptive Metropolis algorithm. The proposal for iteration $n+1$ is $\boldsymbol{Y}_{n+1} \sim N_d(\boldsymbol{X}_n, c\boldsymbol{\Sigma}_n)$, where $\boldsymbol{X}_n$ is the d-dimensional column vector representing the current location of the chain. This proposal is then accepted or rejected according to the usual Metropolis-Hastings ratio. The authors proposed the formula:

$$\boldsymbol{\Sigma}_n = \begin{cases} \boldsymbol{\Sigma}_0 & n \leq n_0 \\ \text{cov}(\boldsymbol{X}_0, \ldots, \boldsymbol{X}_n) + \epsilon \boldsymbol{I}_d & n > n_0, \end{cases} \tag{1}$$

where ϵ is a small positive constant and $\boldsymbol{I}_d$ is the d-dimensional identity matrix. Recall that $\text{cov}(\boldsymbol{X}_0, \ldots, \boldsymbol{X}_n) = \frac{1}{n}\sum_{i=0}^{n} \left(\boldsymbol{X}_i - \overline{\boldsymbol{X}}_n\right)\left(\boldsymbol{X}_i - \overline{\boldsymbol{X}}_n\right)^T$, where $\overline{\boldsymbol{X}}_n = \frac{1}{n+1}\sum_{i=0}^{n} \boldsymbol{X}_i$. The authors also suggested using $c = 2.38^2/d$, which was proved to be optimal for a range of

targets (Gelman, Roberts & Gilks 1996; Roberts, Gelman & Gilks 1997; Roberts & Rosenthal 2001), when $\boldsymbol{\Sigma}_n$ is replaced with the true covariance matrix of the target distribution.

Equation (1) admits the following iterative formulae for calculating the covariance matrix, for $n > n_0$,

$$\overline{\boldsymbol{X}}_n = \frac{n}{n+1}\overline{\boldsymbol{X}}_{n-1} + \frac{1}{n+1}\boldsymbol{X}_{n-1}$$
$$\boldsymbol{\Sigma}_n = \frac{n-1}{n}\boldsymbol{\Sigma}_{n-1} + \frac{1}{n}(\boldsymbol{X}_n\boldsymbol{X}_n^T + n\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T - (n+1)\overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T + \epsilon\boldsymbol{I}_d).$$

The above algorithm becomes increasingly efficient as the covariance matrix adapts to the target. However in the early iterations there can be a couple of major inefficiencies. Firstly, no adaptation occurs at all for the first n_0 iterations and so if the scale of the initial covariance matrix $\boldsymbol{\Sigma}_0$ has been badly chosen then these iterations are completely wasted, leaving nothing on which to base the proposal for subsequent iterations. Secondly, the estimate of the covariance matrix always includes the (arbitrary) starting location of the chain, as well as the following burn-in. Since the usual mean and covariance estimates are sensitive to outliers, it can take a long time for the influence of these points to have reduced enough for the proposal to become efficient. Interestingly, in an earlier paper (Haario, Saksman & Tamminen 1999), the authors describe the Adaptive Proposal algorithm, which uses a fixed number of the most recent observations to estimate the covariance matrix, avoiding this pitfall.

In this paper a more general shaping algorithm is described, termed the Accelerated Shaping algorithm, that includes both the Adaptive Metropolis and Adaptive Proposal algorithms as special cases. Locations visited in the early iterations of the MCMC are removed from the estimate of the covariance matrix, potentially at a rate slower than new ones are accumulated. The framework allows the adaptation to occur smoothly and to begin immediately, avoiding a sharp transition in the proposal. The aim is to make the proposal distribution as effective as possible for every iteration, even in the early stages.

Let $f(n)$ be a non-decreasing sequence of integers such that $f(1) = 0$ and either $f(n+1) = f(n)$ or $f(n+1) = f(n) + 1$ for all n. For example, $f(n) = \lfloor \frac{n}{2} \rfloor$. Consider for $n > 0$,

$$\boldsymbol{\Sigma}_n = w_n \text{cov}(\boldsymbol{X}_{f(n)}, \ldots, \boldsymbol{X}_n) + \boldsymbol{S}_n, \tag{2}$$

where (w_n) is a non-negative sequence and $(\boldsymbol{S}_n)$ is a sequence of positive definite $d \times d$ matrices. In what follows redefine $\overline{\boldsymbol{X}}_n = \frac{1}{n-f(n)+1}\sum_{i=f(n)}^{n}\boldsymbol{X}_i$.

Lemma 1. *If $n > 1$ and $w_{n-1} > 0$ then Equation (2) can be calculated iteratively using the update rules below.*

If $f(n) = f(n-1)$ then a new observation is included:

$$\begin{aligned}\overline{\boldsymbol{X}}_n &= \tfrac{n-f(n)}{n-f(n)+1}\overline{\boldsymbol{X}}_{n-1} + \tfrac{1}{n-f(n)+1}\boldsymbol{X}_n \\ \boldsymbol{\Sigma}_n &= \frac{w_n(n-1-f(n))}{w_{n-1}(n-f(n))}(\boldsymbol{\Sigma}_{n-1} - \boldsymbol{S}_{n-1}) + w_n\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T \\ &\quad + \frac{w_n}{n-f(n)}\left(\boldsymbol{X}_n\boldsymbol{X}_n^T - (n-f(n)+1)\overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T\right) + \boldsymbol{S}_n.\end{aligned}$$

Otherwise $f(n) = f(n-1)+1$ and the new observation replaces the oldest:

$$\begin{aligned}\overline{\boldsymbol{X}}_n &= \overline{\boldsymbol{X}}_{n-1} + \tfrac{1}{n-f(n)+1}\left(\boldsymbol{X}_n - \boldsymbol{X}_{f(n)-1}\right) \\ \boldsymbol{\Sigma}_n &= \frac{w_n}{w_{n-1}}(\boldsymbol{\Sigma}_{n-1} - \boldsymbol{S}_{n-1}) + \frac{w_n}{n-f(n)}\Bigg(\boldsymbol{X}_n\boldsymbol{X}_n^T - \boldsymbol{X}_{f(n)-1}\boldsymbol{X}_{f(n)-1}^T \\ &\quad + (n-f(n)+1)\left[\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T - \overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T\right]\Bigg) + \boldsymbol{S}_n.\end{aligned}$$

Important special cases of Equation (2) include the Adaptive Metropolis algorithm, in which $f(n) \equiv 0$ and if $n \leq n_0$ then $w_n = 0$ and $\boldsymbol{S}_n = \boldsymbol{\Sigma}_0$, whilst if $n > n_0$ then $w_n = 1$ and $\boldsymbol{S}_n = \epsilon\boldsymbol{I}_d$; and the Adaptive Proposal algorithm, in which the covariance estimate is based on the most recent H observations, with $f(n) = \max(0, n-H)$, and for $n \geq H$, $\boldsymbol{S}_n = \mathbf{0}_{d\times d}$ (the $d \times d$ zero matrix) and $w_n = 1$.

2.2. Bayesian learning of the covariance matrix

In this section we discuss learning the covariance matrix of the target in a Bayesian way, and use this idea to find suitable choices for the weight of the empirical covariance matrix, w_n, and the regularizing sequence of covariance matrices, $\boldsymbol{S}_n$.

For the accelerated shaping algorithm to converge towards the optimal proposal then it is required that $n - f(n) \to \infty$, $w_n \to 1$ and $\boldsymbol{S}_n \to \mathbf{0}_{d\times d}$. The convergence and ergodicity properties of the algorithm are discussed further in Section 2.4. To avoid the need to specify the length of the burn-in a priori, it is desirable to have a smooth transition between the initial guess for the covariance estimate $\boldsymbol{\Sigma}_0$ and the current estimate. The aim is for $\boldsymbol{\Sigma}_n$ to represent the current best estimate of the covariance matrix of the target after n iterations, taking into account that in early iterations the empirical covariance estimate is likely to be poor. One solution to this problem is to allow $\boldsymbol{\Sigma}_n$ to be based on a Bayesian analysis of the covariance matrix of the states visited by the MCMC, with the initial covariance matrix $\boldsymbol{\Sigma}_0$ taking the

role of the estimate from the prior. As more observations are collected then the influence of the prior diminishes in a natural way.

Let $\boldsymbol{\mu}$ denote the mean and $\boldsymbol{\Sigma}$ denote the covariance matrix of the target distribution being explored by the MCMC. Assume a normal inverse-Wishart distribution for $(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \sim \text{NIW}(\boldsymbol{\mu}_0, \eta_0, \boldsymbol{C}_0, \nu_0)$, which has density (Prince 2012)

$$\pi(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{\mu}_0, \eta_0, \boldsymbol{C}_0, \nu_0) = \frac{\eta_0^{d/2}|\boldsymbol{C}_0|^{\nu_0/2}}{2^{\nu_0 d/2}(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{\nu_0+d+2}\Gamma_d(\nu_0/2)} \times \exp\left\{-\frac{1}{2}\left(\text{Tr}(\boldsymbol{C}_0\boldsymbol{\Sigma}^{-1}) + \eta_0(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_0)\right)\right\}.$$

A posterior for $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is going to be derived, based on the states visited between iterations $f(n)$ and n, as in Section 2.1, and so the analogous notation will be used. Suppose that $n - f(n) + 1$ independent observations of data are observed from a multivariate normal $\boldsymbol{X}_i|(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \sim N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with labels $i = f(n), \ldots, n$. Using the conjugacy of the normal inverse-Wishart prior, the posterior is $(\boldsymbol{\mu}, \boldsymbol{\Sigma})|[\boldsymbol{X}_i]_{i=f(n)}^n \sim \text{NIW}(\boldsymbol{\mu}_n, \eta_n, \boldsymbol{C}_n, \nu_n)$, where

$$\begin{aligned}
\boldsymbol{\mu}_n &= \frac{\eta_0\boldsymbol{\mu}_0 + (n-f(n)+1)\overline{\boldsymbol{X}}_n}{\eta_0 + n - f(n) + 1} \\
\eta_n &= \eta_0 + n - f(n) + 1 \\
\boldsymbol{C}_n &= \boldsymbol{C}_0 + (n-f(n))\,\text{cov}(\boldsymbol{X}_{f(n)}, \ldots, \boldsymbol{X}_n) + \tfrac{\eta_0(n-f(n)+1)}{\eta_0+n-f(n)+1}(\overline{\boldsymbol{X}}_n - \boldsymbol{\mu}_0)(\overline{\boldsymbol{X}}_n - \boldsymbol{\mu}_0)^T \\
\nu_n &= \nu_0 + n - f(n) + 1,
\end{aligned}$$

where $\overline{\boldsymbol{X}}_n = \frac{1}{n-f(n)+1}\sum_{i=f(n)}^{n}\boldsymbol{X}_i$ as before.

The parameters η_0 and ν_0 quantify the strength of the prior information for the mean and covariance matrix respectively. From the updating rules above, it is possible to interpret these parameters in terms of the number of observations of data that is represented in the prior. Under this model then the maximum a posteriori (MAP) estimator for the covariance matrix is $\boldsymbol{\Sigma}_n = \frac{1}{n-f(n)+\nu_0+d+2}\boldsymbol{C}_n$. Choosing the prior covariance matrix so that the prior mode is equal to the initial proposal covariance matrix $\boldsymbol{\Sigma}_0$ yields $\boldsymbol{C}_0 = (\nu_0 + d + 1)\boldsymbol{\Sigma}_0$. Finally, to reduce the number of algorithmic parameters and to avoid the need to specify a prior for the mean of the target, $\boldsymbol{\mu}_0$, it is possible to set $\eta_0 = 0$.

This leads to

$$\begin{aligned}
w_n &= \frac{n-f(n)}{n-f(n)+\nu_0+d+2} \\
\boldsymbol{S}_n &= \frac{\nu_0+d+1}{n-f(n)+\nu_0+d+2}\boldsymbol{\Sigma}_0.
\end{aligned}$$

These assumptions produce some cancellations in Lemma 1. For $f(n) = f(n-1)$,

$$\boldsymbol{\Sigma}_n = \tfrac{1}{n-f(n)+\nu_0+d+2}\Big((n-f(n)+\nu_0+d+1)\boldsymbol{\Sigma}_{n-1} + \boldsymbol{X}_n\boldsymbol{X}_n^T + (n-f(n))\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T - (n-f(n)+1)\overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T\Big); \quad (3)$$

and for $f(n) = f(n-1)+1$

$$\boldsymbol{\Sigma}_n = \boldsymbol{\Sigma}_{n-1} + \tfrac{1}{n-f(n)+\nu_0+d+2}\Big(\boldsymbol{X}_n\boldsymbol{X}_n^T - \boldsymbol{X}_{f(n)-1}\boldsymbol{X}_{f(n)-1}^T + (n-f(n)+1)(\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T - \overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T)\Big), \quad (4)$$

where $n > 1$ in both cases. In general the observations from the MCMC will be neither independent nor distributed according to the multivariate normal distribution, however nonetheless the motivation above can still be used to justify the posterior mode as the best available estimate of the covariance matrix of the target.

2.3. Accelerated scaling

In Algorithm 4 of Andrieu & Thoms (2008), the authors suggest changing the global scale of the covariance matrix in order to achieve a target acceptance rate. This modifies the proposal distribution to $\boldsymbol{Y}_{n+1} \sim N_d(\boldsymbol{X}_n, \lambda_n^2 c\boldsymbol{\Sigma}_n)$. A good choice (Atchadé & Rosenthal 2005; Andrieu & Thoms 2008; Garthwaite, Fan & Sisson 2016) for adapting the global scale parameter λ_n is to use the stochastic search algorithm known as the Robbins-Munro algorithm (Robbins & Monro 1951). This algorithm can be used to find the solution to the equation $p(\lambda) = a$ for some monotonically decreasing function of λ, based on the results of some Bernoulli random variables with success probability $p(\lambda)$. Let $\theta_n = \log(\lambda_n)$, then θ_n is updated iteratively via

$$\theta_{n+1} = \begin{cases} \theta_n + \frac{\delta}{n}(1-a) & \text{if the } n\text{th trial is a success,} \\ \theta_n - \frac{\delta}{n}a & \text{if the } n\text{th trial is a failure.} \end{cases}$$

When applied to the Metropolis-Hastings random walk algorithm, this adaptive scheme leads to the scale increasing after each accepted proposal and reducing after each rejected proposal, so that the expected increase is zero if and only if the success probability is a. Alternatively, it is possible to remove the randomness introduced by the Bernoulli trials by setting

$$\theta_{n+1} = \theta_n + \frac{\delta}{n}(\alpha(\boldsymbol{Y}_{n+1}|\boldsymbol{X}_n) - a), \quad (5)$$

where $\alpha(\boldsymbol{Y}_{n+1}|\boldsymbol{X}_n)$ is the Metropolis-Hastings acceptance probability for proposal $\boldsymbol{Y}_{n+1}$.

In Garthwaite, Fan & Sisson (2016) this algorithm is refined for adaptive MCMC. First the authors derive a suitable step size constant $\delta = \left(1 - \frac{1}{d}\right)\left(\frac{\sqrt{2\pi}\exp(A^2/2)}{2A}\right) + \frac{1}{da(1-a)}$, where $A = -\Psi^{-1}(a/2)$ and Ψ is the cdf of a standard normal distribution. Second, the authors introduce a check to prevent the algorithm diminishing too rapidly and being unable to reach the target. If λ_n changes by a factor of 3 from its starting value (or equivalently $|\theta_n - \theta_0| > \log(3)$) then the algorithm is restarted from its current location. Finally, they begin (or restart) the algorithm at $n = \mathrm{round}(5/(a(1-a))$ to avoid rapid changes in scale in the early stages, or after a restart. These changes are summarised in Algorithm 1 using a slightly different notation to avoid the confusion of having more than one iteration with index n. In this revised notation it is clear that the rounding of the starting iteration is unnecessary.

As the dimension of the target increases, the optimal acceptance rate has been shown to rapidly approach 0.234 (Gelman, Roberts & Gilks 1996; Roberts & Rosenthal 2001), at least when the target is Gaussian or can be written as a product over its dimensions, ie $\pi(\boldsymbol{x}) = \prod_{i=1}^{d} \pi_i(x_i)$. For such targets we would expect to achieve the optimal acceptance rate when $\lambda_n = 1$, giving the optimal scaling discussed previously. However, if the covariance matrix of the target is underestimated in the early iterations of an adaptive algorithm, the scale can be inflated to keep 'pushing at the boundaries' of the target in order to increase the rate at which the true covariance matrix is estimated. Conversely, if the covariance matrix is overestimated (perhaps due to some outlying points from the burn-in), then reducing the scale of the target prevents the acceptance rate becoming too small and the chain getting stuck. Reducing the scale by too much can cause very slow mixing and so to prevent this it is sometimes necessary to impose a minimum value for λ_n, for example $\lambda_{\min} = 1$ which corresponds to $c\lambda_n^2 = 2.38^2/d$. If this value is reached then the algorithm will not achieve the target acceptance rate a. This version of the Robbins-Munro scaling algorithm is referred to here as the accelerated scaling algorithm.

2.4. Convergence and ergodicity

This section discusses some conditions for the proposed algorithms to converge to the correct target. Since the accelerated shaping algorithm includes both the adaptive Metropolis algorithm (which is ergodic for the target π) and the adaptive proposal algorithm (which is not always ergodic for π) as special cases, then it is clear that further conditions must be introduced to imply the correct ergodicity. Finally, the ways in which the algorithm can fail to converge are discussed, which suggest some additional checks for convergence.

Roberts & Rosenthal (2007) introduces two conditions that imply convergence and the correct ergodicity for adaptive MCMC algorithms. The authors consider a collection of Markov chain kernels $\{P_\gamma\}_{\gamma \in \mathcal{Y}}$ on a state space $\mathcal{X}$, each of which has stationary distribution

Algorithm 1: Accelerated scaling and shaping algorithm

inputs: $\boldsymbol{X}_0, \boldsymbol{\Sigma}_0, a, \nu_0, f(\cdot), \lambda_{\min}, N$

$d = \text{dimension}(\boldsymbol{X}_0)$ // initialisation

$c = 2.38^2/d$

$\lambda_{\text{start}} = \lambda_0 = 1$

$n_{\text{start}} = 5/a(1-a)$

$A = -\Psi^{-1}(a/2)$

$\delta = \left(1 - \frac{1}{d}\right)\left(\frac{\sqrt{2\pi}\exp(A^2/2)}{2A}\right) + \frac{1}{da(1-a)}$

for $n = 1, \ldots, N$ **do**

$\boldsymbol{Y}_n \sim N_d(\boldsymbol{X}_{n-1}, \lambda_{n-1} c \boldsymbol{\Sigma}_{n-1})$

$u \sim U(0,1)$

if $u < \alpha(\boldsymbol{Y}_n | \boldsymbol{X}_{n-1})$ **then** $\boldsymbol{X}_n = \boldsymbol{Y}_n$ // accept

else $\boldsymbol{X}_n = \boldsymbol{X}_{n-1}$ // reject

if $n = 1$ **then**

// initialise estimates

$$\overline{\boldsymbol{X}}_1 = \tfrac{1}{2}\sum_{i=0}^{1} \boldsymbol{X}_i$$

$$\boldsymbol{\Sigma}_1 = \tfrac{1}{\nu_0+d+3}\left(\sum_{i=0}^{1} \boldsymbol{X}_i \boldsymbol{X}_i^T - 2\overline{\boldsymbol{X}}_1 \overline{\boldsymbol{X}}_1^T + (\nu_0 + d + 1)\boldsymbol{\Sigma}_0\right)$$

else if $f(n) = f(n-1)$ **then**

// include new observation

$$\overline{\boldsymbol{X}}_n = \tfrac{n-f(n)}{n-f(n)+1}\overline{\boldsymbol{X}}_{n-1} + \tfrac{1}{n-f(n)+1}\boldsymbol{X}_n$$

$$\boldsymbol{\Sigma}_n = \tfrac{1}{n-f(n)+\nu_0+d+2}\Big((n-f(n)+\nu_0+d+1)\boldsymbol{\Sigma}_{n-1} + \boldsymbol{X}_n\boldsymbol{X}_n^T + (n-f(n))\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T - (n-f(n)+1)\overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T\Big)$$

else if $f(n) = f(n-1) + 1$ **then**

// new observation replaces oldest

$$\overline{\boldsymbol{X}}_n = \overline{\boldsymbol{X}}_{n-1} + \tfrac{1}{n-f(n)+1}\left(\boldsymbol{X}_n - \boldsymbol{X}_{f(n)-1}\right)$$

$$\boldsymbol{\Sigma}_n = \boldsymbol{\Sigma}_{n-1} + \tfrac{1}{n-f(n)+\nu_0+d+2}\Big(\boldsymbol{X}_n\boldsymbol{X}_n^T - \boldsymbol{X}_{f(n)-1}\boldsymbol{X}_{f(n)-1}^T + (n-f(n)+1)(\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T - \overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T)\Big)$$

$\lambda_n = \max\left\{\lambda_{\min}, \lambda_{n-1}\exp\left(\frac{\delta}{n_{\text{start}}+n}(\alpha(\boldsymbol{Y}_n|\boldsymbol{X}_{n-1}) - a)\right)\right\}$ // update scale

if $|\log(\lambda_n) - \log(\lambda_{\text{start}})| > \log(3)$ **then**

$\lambda_{\text{start}} \mapsto \lambda_n$ // restart Robbins-Munro

$n_{\text{start}} \mapsto 5/a(1-a) - n$

equal to the target, π. An adaptive algorithm is then given by a sequence of states visited by the Markov chain $(X_n)_n$ along with a sequence of $\mathcal{Y}$-valued random variables $(\Gamma_n)_n$ that indicate the choice of kernal at iteration $n+1$. The first condition, which is needed for the

correct ergodicity, is *diminishing adaptation*,

$$\lim_{n\to\infty} \sup_{x\in\mathcal{X}} ||P_{\Gamma_{n+1}}(x,\cdot) - P_{\Gamma_n}(x,\cdot)|| = 0 \text{ in probability,} \tag{6}$$

where $||\cdot||$ denotes the total variation norm. The second condition is *containment*, which states that the convergence times are bounded in probability, ie $\{M_\epsilon(X_n,\Gamma_n)\}_{n=1}^\infty$ is bounded in probability, where

$$M_\epsilon(x,\gamma) = \inf\{n \geq 1 : ||P_\gamma^n(x,\cdot) - \pi(\cdot)|| \leq \epsilon\}. \tag{7}$$

The conditions are not satisfied by accelerated shaping or accelerated scaling in general. Previous authors have introduced additional conditions, for example that $\mathcal{X}$ is bounded (Haario et al. 2001); that $\boldsymbol{\Sigma}_n$ cannot shrink to zero (eg $\boldsymbol{S}_n = \epsilon \boldsymbol{I}_d$, Haario et al. 2001); or that each iteration may be drawn from a fixed kernel with small probability (Roberts & Rosenthal 2009) to prevent the chain getting stuck. Although these additional conditions facilitate the proof of ergodicity to the correct target, they make the algorithm less efficient than the optimal non-adaptive algorithm. These differences can be made to be small, but some parameters may be difficult for inexperienced practitioners to interpret. For example if the scales of the parameters are radically different then choosing appropriate values of ϵ, when adding $\epsilon \boldsymbol{I}_d$ to the proposal covariance matrix, may be challenging.

Craiu et al. (2015) prove that, under certain technical conditions, an adaptive MCMC algorithm converges to the correct target in total variation distance; as long as adaptation only occurs within a compact subset $K \subseteq \mathcal{Y}$, with a fixed and bounded proposal outside of K. This implies that a suitable adaptive algorithm that remains within a compact subset of $\mathcal{Y}$ must converge to the correct target.

Vihola (2011) considers two adaptive Metropolis-Hastings algorithms – one with adaptive scaling using Robbins Munro updates and one with adaptive scaling and shaping that are very close in spirit to the algorithms discussed in this paper. Under either assumptions 1 and 2 below; or assumptions 1 and 3; a strong law of large numbers is proved for bounded functions f on the target π for both algorithms, ie.

$$\frac{1}{n}\sum_{k=1}^{n} f(\boldsymbol{X}_k) \to \int_{\mathbb{R}^d} f(\boldsymbol{x})\pi(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \text{ almost surely as } n \to \infty.$$

1. There exists a collection of contour sets of the target π with uniformly continuous normals (see Definition 6 in Vihola 2011).
2. The target π has compact support.

3. The target π is bounded, bounded away from zero on compact sets, differentiable and has super-exponentially decaying tails (see Assumption 8 of Vihola 2011).

Although the scaling constant is not restricted to a compact set, it must be bounded away from zero. Furthermore the covariance matrix of the proposal is restricted to reside within a compact set, by assuming that adaptation cannot occur unless the eigenvalues of the unscaled proposal covariance matrix remain within $[\zeta^{-1}, \zeta]$ for some $\zeta \in [1, \infty)$.

[relate to my algorithm]

Furthermore, conditions that ensure asymptotic convergence do not guarantee that a finite sample from such an MCMC chain will resemble a sample from the target, as is always the case with MCMC, and may give a false sense of security.

Instead, the following discussion describes three ways in which the accelerated algorithms can fail to converge, along with some suggested methods to identify when this has occurred. First, if the accelerated shaping algorithm gets stuck in a location then the empirical estimate of the covariance matrix will shrink to the zero matrix, which can lead to $\mathbf{\Sigma}_n$ approaching the zero matrix. This will be obvious from the very low acceptance rate and in a trace plot of the entries of $\mathbf{\Sigma}_n$. Second, the accelerated scaling algorithm can fail to converge if λ_n tends to zero, which can be identified from a trace plot of $\log(\lambda_n)$. Finally, if the target has very heavy tails then the scale of the proposal may continue increasing, possibly indefinitely. This can be identified from a trace plot of λ_n. For such targets the MHRW algorithm is not a good choice and is unlikely to be successful under any kind of adaptation. Practical solutions might include putting bounds on the parameters or developing an informative prior for the parameters.

3. Results

3.1. Accelerated shaping example

A simple 2-dimensional example shows that removing as well as adding observations to the estimation of the covariance matrix speeds up the time taken to obtain a reasonable estimate when the chain is not started close to the posterior mode. The example target is a multivariate normal with mean $(0, 200)$ and covariance matrix $\mathbf{\Sigma} = [50, -40; -40, 50]$ – an elliptical ridge with strong negative correlation. If the chains are started at $(0, 0)$ then the initial gradient sends the chain into the positive quadrant tricking the covariance estimate into having positive correlation. Once a chain reaches the crest of the ridge it must change direction and follow the ridge-line up to the summit, going against it's fledgling correlation estimate. Having reached the summit, the chain must then forget the burn-in and learn the true covariance matrix.

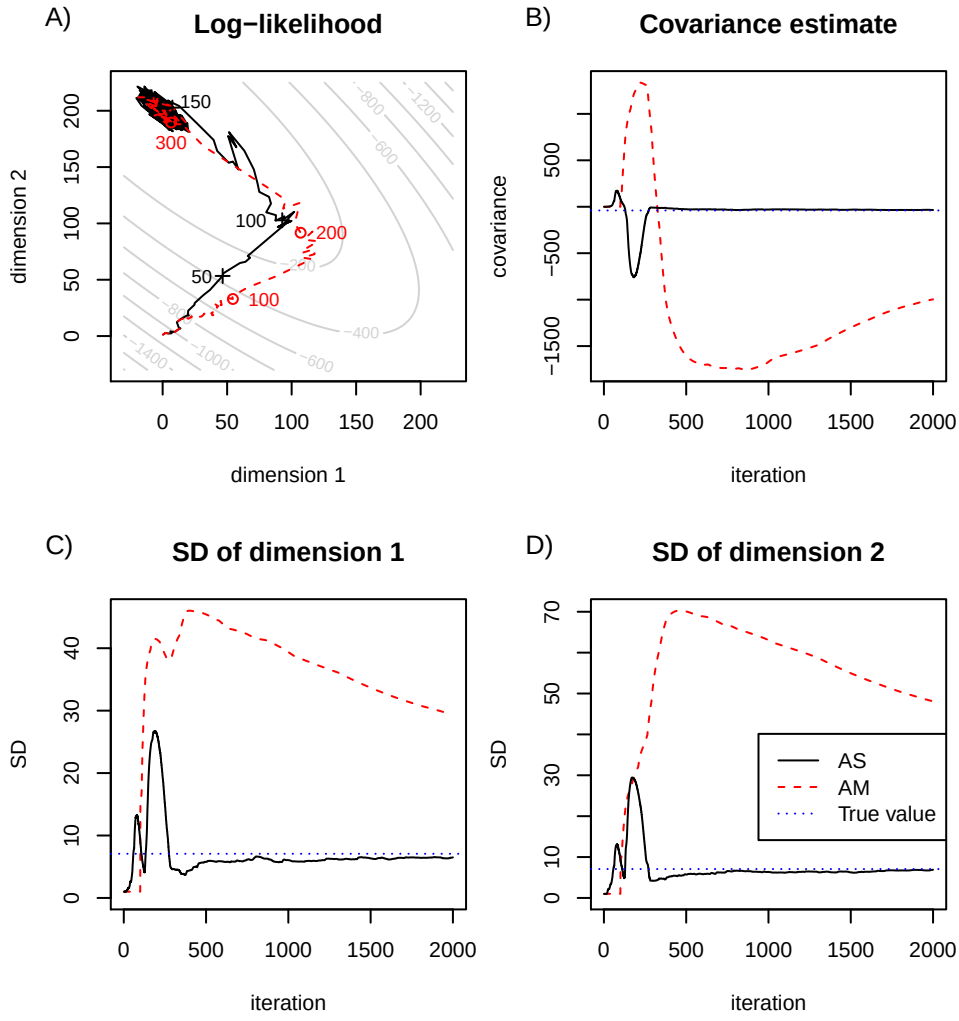

Figure 1. Comparison of Accelerated Shaping (AS) and Adaptive Metropolis (AM) algorithms. A) Contour plot of the log-posterior with traces, B) running covariance estimate, and C) and D) running standard deviation (SD) estimates for AS (black) and AM (red, dashed).

We compared the performance of the Adaptive Metropolis algorithm (Haario et al. 2001) with the accelerated shaping algorithm described in Section 2.1. A contour plot showing the traces and running estimates of the entries of the proposal covariance matrix are shown in Figure 1. Both algorithms used $\boldsymbol{\Sigma}_0 = \boldsymbol{I}_2$, the Adaptive Metropolis algorithm used $n_0 = 100$ and $\epsilon = 0.01$; whilst the accelerated shaping algorithm used $\nu_0 = 100$. The accelerated shaping algorithm in equations (3) and (4) is seen to converge towards the true $\boldsymbol{\Sigma}$ much more rapidly and the trace reaches the posterior mode approximately twice as fast. In this toy

2-dimensional example the chains were started a long way from the mode to emphasize the difference between the two algorithms. This was intended to mimic more realistic problems in higher dimensions, in which a starting location close to the mode becomes hard to identify a priori.

3.2. Global scaling example

For multivariate normal targets or targets that can be written in product form (ie $\pi(\boldsymbol{x}) = \prod_{i=1}^{d} \pi_i(x_i)$), the optimal scale of the proposal is achieved with a scaling constant of $2.38^2/d$, which yields an acceptance rate of 0.234. But if the target does not satisfy the required conditions, then an algorithm can be tuned to either one of these at the expense of the other. But which criteria should be aimed for?

Consider the 'banana-shaped' target (Roberts & Rosenthal 2009; Haario, Saksman & Tamminen 1999) with density

$$f(x_1, \ldots, x_d) \propto \exp\{-x_1^2/200 - \tfrac{1}{2}(x_2 + Bx_1^2 - 100B)^2 - \tfrac{1}{2}(x_3^2, \ldots, x_d^2)\},$$

with 'bananicity constant' $B = 0.1$ and dimension $d = 2$. Robbins-Munro adaptive scaling algorithms with a range of target acceptance rates were compared with a non-adaptive algorithm with $\lambda_n \equiv 1$. In all cases the covariance matrix was assumed known, i.e. $\boldsymbol{\Sigma}_n = \boldsymbol{\Sigma}$ for all n. For the adaptive algorithms, the scaling constant λ_n was started from $\lambda_1 = 1$ and updated via Equation (5), recaling that $\theta_n = \log(\lambda_n)$. Here, λ_{min} was set to 0 for this example to illustrate the effect on the mixing of shrinking the proposal below $\lambda_n = 1$. All chains were started at the posterior mode and run for 2×10^5 iterations.

Table 1 shows the mean squared jumping distance (MSJD), the mean Euclidean jumping distance (MEJD) and minimum effective sample size (ESS) for each algorithm, (for definitions see Sherlock, Fearnhead & Roberts 2010). The effective sample size was calculated using the function `effectiveSize` from the R package `coda` (Plummer et al. 2006).

Table 1 shows that $\lambda_n < 1$ when the target acceptance rate was $a = 0.234$. Approximately the largest MSJD (which equates to the best mixing) and ESS were produced by setting $\lambda_n \equiv 1$. However, targeting 0.234 gave approximately the best MEJD (distance travelled). As an aside, the optimal acceptance rate would be higher than 0.234 if the conditions of the theorem held since this target has just 2 dimensions. A secondary observation from Table 1 is that the Robbins-Munro algorithm is generally accurate in achieving the desired acceptance rate.

This simple example showed that shrinking the proposal to achieve the 'optimal' acceptance rate actually made the mixing worse. However, there are still possible advantages

a	$\widehat{a}$	$\overline{\lambda}$	MSJD	MEJD	ESS
0.01	0.01	2.04	0.0343	0.17	786
0.02	0.0199	1.33	0.0493	0.29	1254
0.03	0.03	0.98	0.0551	0.38	727
0.05	0.0488	0.68	0.0513	0.47	726
0.1	0.0984	0.37	0.0348	0.56	610
0.2	0.1899	0.19	0.0202	0.61	386
0.234	0.2213	0.16	0.0176	0.62	315
0.3	0.2858	0.12	0.0136	0.63	246
0.4	0.3936	0.08	0.0094	0.63	163
0.45	0.4466	0.07	0.0074	0.6	117
$\lambda_n \equiv 1$	0.0296	1	0.0548	0.38	1177

Table 1. Mean squared jumping distance (MSJD), mean Euclidean jumping distance (MEJD) and effective sample size (ESS) for 2×10^5 iterations of the accelerated scaling algorithm with 2-dimensional Banana-shaped target with $B = 0.1$. Normal proposals using the true covariance matrix were scaled by λ_n using the Robbins-Munro adaptive scaling algorithm in order to target an acceptance rate of a. Observed acceptance rates are given by $\widehat{a}$ and the mean value of λ_n is given by $\overline{\lambda}$.

of rescaling a multivariate proposal, for example maximising the distance travelled during the burn-in may help an adaptive algorithm to estimate the covariance matrix of the target more rapidly. This will be especially true when the covariance matrix is underestimated in at least some dimensions. Shrinking the proposal also increases the acceptance rate and prevents the chain from getting stuck. In light of this example, in future the Robbins-Munro algorithm will be prevented from shrinking the proposal below $c_n = 2.38^2/d$. This can easily be achieved by setting $\lambda_{\text{min}} = 1$, and replacing equation (5) with $\theta_{n+1} = \max\{\log(\lambda_{\text{min}}), \theta_n + \frac{\delta}{n}(\alpha(\boldsymbol{Y}_{n+1}|\boldsymbol{X}_n) - a)\}$ in the Robbins-Munro update. This version of the Robbins-Munro algorithm will be referred to as the accelerated scaling algorithm.

3.3. Choosing the forgetting sequence, $f(n)$

The final aspect of Algorithm 1 that needs to be determined is an appropriate choice of the forgetting sequence, $f(n)$. To determine this, consider another example from Roberts & Rosenthal (2009) with a higher dimensional target, where there is a great deal more learning to be done in the proposal covariance matrix. Let $\boldsymbol{M}$ be a $d \times d$ matrix with entries drawn from independent standard normal distributions, and then form $\boldsymbol{\Sigma} = \boldsymbol{M}\boldsymbol{M}^T$. The target is then the d dimensional multivariate normal with mean zero and covariance matrix $\boldsymbol{\Sigma}$.

The first phase of an adaptive MCMC algorithm is the *transient* phase, where the chain travels towards the high posterior mass of the target distribution. Once the states from the transient phase have been removed from the estimate of the covariance matrix there is no need to remove further observations, so this example will concentrate on finding the best

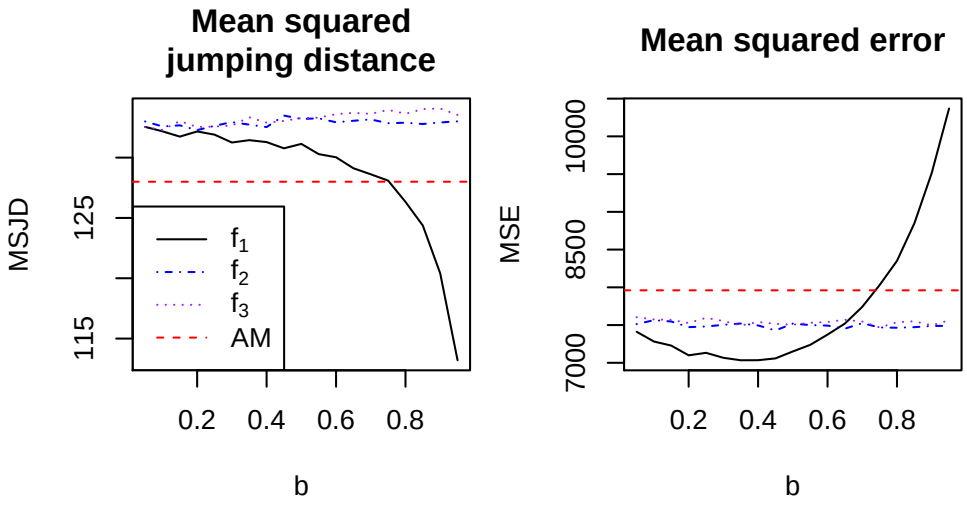

Figure 2. Varying the forgetting sequence $f(n)$ for Gaussian targets in 20 dimensions. Comparison between Adaptive Metropolis algorithm (AM) with $n_0 = 20$ and $\epsilon = 0.01$ and the accelerated shaping and scaling algorithm with $\nu_0 = 20$ and forgetting sequences $f_1(n) = \lfloor bn \rfloor$, $f_2(n) = \lfloor b\sqrt{2n} \rfloor$ and $f_3(n) = \lfloor b \log(n) \times 2/\log(2) \rfloor$. Chains are run for 10000 iterations and results are averages over 100 randomly chosen targets.

sequence $f(n)$ for forgetting the transient phase. Three functional forms for $f(n)$ will be considered, for $0 < b < 1$:

1. $f_1(n) = \lfloor bn \rfloor$.
2. $f_2(n) = \lfloor b\sqrt{2n} \rfloor$.
3. $f_3(n) = \lfloor b \log(n) \times 2/\log(2) \rfloor$.

Here, the constants have been chosen to ensure $f_i(2) = 1$ whilst maximising the impact of varying $b \in (0, 1)$.

Figure 2 shows the results from the three forgetting sequences as a function of the scaling constant b. For comparison, the Adaptive Metropolis algorithm is also shown. Chains were started at $X_0 = (5, 0, \ldots, 0)$, which is away from the posterior mode at the origin, and run for 10,000 iterations. The results shown are averages over 100 randomly generated Gaussian targets with dimension $d = 20$. The results show that the linear function f_1 achieves the lowest mean square error for values of b around 0.4, however this does not produce the largest mean squared jumping distance. The mean squared jumping distance is highest for f_3 with b close to one. All three functions outperform the adaptive Metropolis (AM) algorithm in both measures, except for f_1 with b close to one. The difference between the AM algorithm and the alternatives with b close to zero is explained by the improved use of the initial iterations, as described in Section 2.2. In conclusion, the performance was not very sensitive

to the choice of b in the interval $(0.2, 0.5)$. Generally b should be chosen to be as small as possible so that once the observations from the transient phase have been removed, as many of the subsequent observations as possible will contribute to the estimate of the covariance matrix. Unfortunately it is impossible to say a priori when the transient phase will end, but if it is expected to be over by iteration B, then an improved forgetting function would be $f(n) = \max\{B, \lfloor bn \rfloor\}$.

3.4. High dimensional example

Finally, consider a high dimensional Gaussian target with $d = 100$, as in Section 3.3. Four adaptive random walk algorithms will be compared: the adaptive Metropolis algorithm, the accelerated shaping algorithm from Section 2.1, the accelerated scaling algorithm (the Robbins-Munro algorithm with $\lambda_{\text{min}} = 1$) and the accelerated shaping and scaling algorithm described in Algorithm 1. The algorithm parameters were $\epsilon = 0.01$ and $n_0 = 100$ for adaptive Metropolis; $\nu_0 = 100$ and $f(n) = \lfloor 0.3n \rfloor$ for accelerated shaping; $a = 0.234$ for accelerated scaling and $\boldsymbol{\Sigma}_0 = \boldsymbol{I}_d$ for all algorithms. Each chain was run for 5×10^5 iterations starting, away from the posterior mode, at $(5, 0, \ldots, 0)$.

Figure 3 shows the mean squared error in the estimate of the covariance matrix, the mean squared jumping distance and the scaling factor λ_n as a function of iteration for each of the four algorithms. Unsurprisingly, it takes the adaptive Metropolis algorithm a considerable number of iterations to learn the 5050 parameters of the covariance matrix. The accelerated scaling algorithm estimated the covariance matrix more rapidly than adaptive Metropolis because it made larger jumps and was able to explore the target more effectively. The accelerated shaping algorithm produced a better covariance estimate than adaptive Metropolis as it used the early iterations more effectively and gradually removed the initial transient phase from the covariance estimate. However, using both accelerated shaping and scaling together produced the lowest mean squared error in the estimate of the covariance matrix and led to the highest mean squared jumping distance. However, as the estimate of the covariance matrix became more accurate, the scaling factor λ_n naturally adapted towards the optimal value of one.

4. Discussion and conclusion

This numerical study has explored two ideas for increasing the rate of adaptation during the early stages of an adaptive Metropolis-Hastings random walk. First, the shaping algorithm of Haario et al. (2001) was modified to adapt smoothly between the initial covariance matrix $\boldsymbol{\Sigma}_0$ and the current estimate $\boldsymbol{\Sigma}_n$; and to remove early outlying locations from the estimate of the covariance matrix at a rate slower than new observations arrive. This was shown to greatly

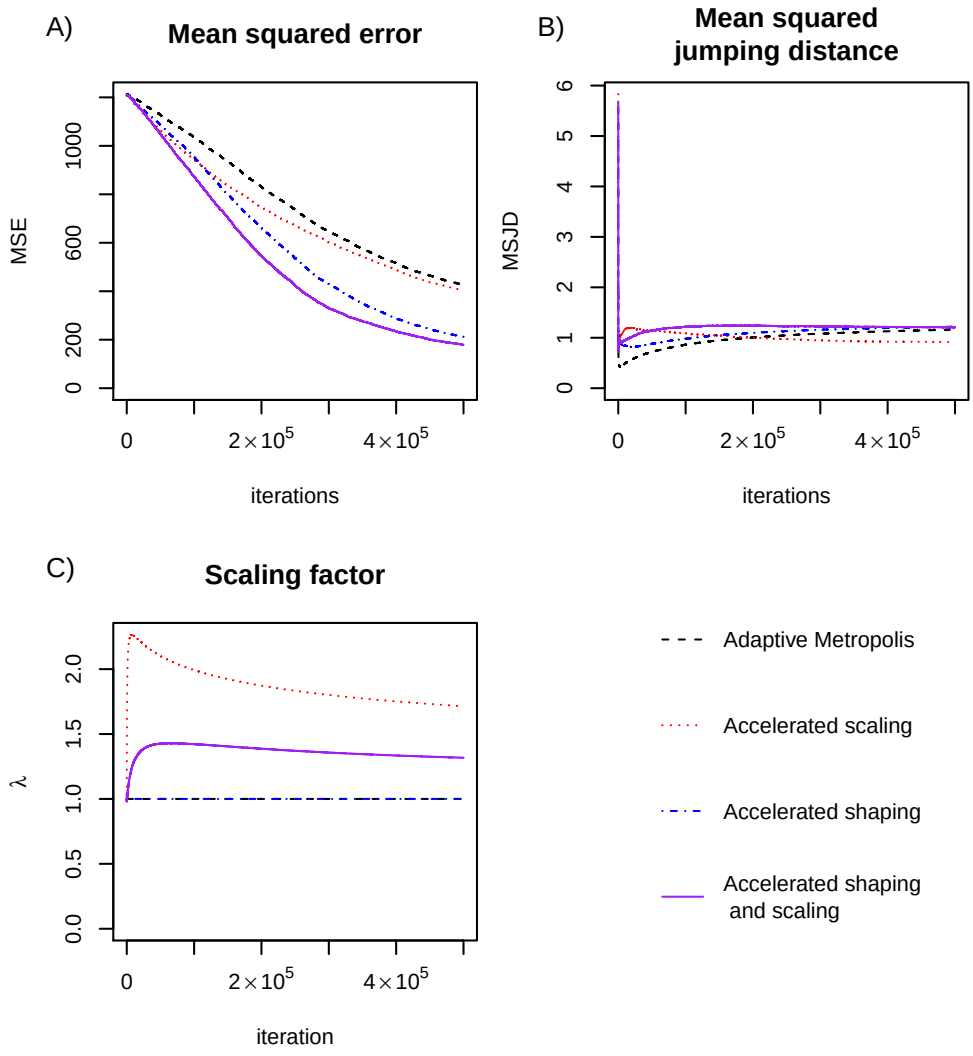

Figure 3. The effect of scaling on the Adaptive Metropolis algorithm for a 100-dimensional multivariate normal target. Plot A shows the total mean squared error in the estimate of the covariance matrix; B shows the mean squared jumping distance and the scaling factor λ_n is shown in C.

improve the rate of convergence to the true covariance matrix. Although this modification is not required if the chain is initialised at the posterior mode, there are circumstances in which the mode can be challenging to obtain, such as when the gradients of the target are not available.

Second, the study explored an approach to scaling the proposal to achieve a target acceptance rate, via the Robbins-Munro algorithm. Although shrinking the proposal turned out to be deleterious in a multivariate banana-shaped example, the accelerated scaling

approach was easily modified to prevent shrinking from occurring. The accelerated scaling and shaping approaches together was shown to increase the rate at which the covariance matrix of the target was learned in a high-dimensional example.

The adaptive Metropolis-Hastings random walk algorithm has an enduring popularity despite the availability of more sophisticated alternatives, largely due to the simplicity of its implementation, wide applicability and robustness to misspecification of algorithmic parameters such as the initial covariance matrix $\boldsymbol{\Sigma}_0$. The modifications described here have been shown to improve the learning rate of the adaptation in unimodel targets and have a negligible cost in terms of increased complexity and difficulty in implementation.

Appendix I

Proof of Lemma 1

First note that

$$\begin{aligned}\boldsymbol{\Sigma}_n &= w_n \text{cov}(\boldsymbol{X}_{f(n)}, \ldots, \boldsymbol{X}_n) + \boldsymbol{S}_n \\ \Rightarrow \text{cov}(\boldsymbol{X}_{f(n-1)}, \ldots, \boldsymbol{X}_{n-1}) &= \frac{1}{w_{n-1}}(\boldsymbol{\Sigma}_{n-1} - \boldsymbol{S}_{n-1}).\end{aligned}$$

Second, recall that

$$\begin{aligned}\text{cov}(\boldsymbol{X}_{f(n)}, \ldots, \boldsymbol{X}_n) &= \frac{1}{n - f(n)} \sum_{i=f(n)}^{n} (\boldsymbol{X}_i - \overline{\boldsymbol{X}}_n)(\boldsymbol{X}_i - \overline{\boldsymbol{X}}_n)^T \\ &= \frac{1}{n - f(n)} \left(\sum_{i=f(n)}^{n} \boldsymbol{X}_i \boldsymbol{X}_i^T \right) - \frac{n - f(n) + 1}{n - f(n)} \overline{\boldsymbol{X}}_n \overline{\boldsymbol{X}}_n^T.\end{aligned}$$

If $f(n) = f(n-1)$ then the latest observation is combined with the previous estimates.

$$
\begin{aligned}
\overline{\boldsymbol{X}}_n &= \frac{1}{n-f(n)+1}\sum_{i=f(n)}^{n}\boldsymbol{X}_i \\
&= \frac{n-f(n)}{n-f(n)+1}\frac{1}{n-f(n)}\sum_{i=f(n-1)}^{n-1}\boldsymbol{X}_i + \frac{1}{n-f(n)+1}\boldsymbol{X}_n \\
&= \frac{n-f(n)}{n-f(n)+1}\overline{\boldsymbol{X}}_{n-1} + \frac{1}{n-f(n)+1}\boldsymbol{X}_n \\
\boldsymbol{\Sigma}_n &= \frac{w_n}{n-f(n)}\left(\sum_{i=f(n)}^{n}\boldsymbol{X}_i\boldsymbol{X}_i^T - (n-f(n)+1)\overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T\right) + \boldsymbol{S}_n \\
&= \frac{w_n}{n-f(n)}\Bigg(\sum_{i=f(n-1)}^{n-1}\boldsymbol{X}_i\boldsymbol{X}_i^T - (n-f(n-1))\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T + (n-f(n-1))\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T \\
&\qquad + \boldsymbol{X}_n\boldsymbol{X}_n^T - (n-f(n)+1)\overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T\Bigg) + \boldsymbol{S}_n \\
&= \frac{w_n}{n-f(n)}\Bigg((n-1-f(n-1))\mathrm{cov}(\boldsymbol{X}_{f(n-1)},\ldots,\boldsymbol{X}_{n-1}) + (n-f(n))\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T \\
&\qquad + \boldsymbol{X}_n\boldsymbol{X}_n^T - (n-f(n)+1)\overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T\Bigg) + \boldsymbol{S}_n \\
&= \frac{w_n}{n-f(n)}\Bigg(\frac{n-1-f(n)}{w_{n-1}}(\boldsymbol{\Sigma}_{n-1}-\boldsymbol{S}_{n-1}) + (n-f(n))\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T \\
&\qquad + \boldsymbol{X}_n\boldsymbol{X}_n^T - (n-f(n)+1)\overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T\Bigg) + \boldsymbol{S}_n \\
&= \frac{w_n(n-1-f(n))}{w_{n-1}(n-f(n))}(\boldsymbol{\Sigma}_{n-1}-\boldsymbol{S}_{n-1}) + w_n\overline{\boldsymbol{X}}_{n-1}\overline{\boldsymbol{X}}_{n-1}^T \\
&\quad + \frac{w_n}{n-f(n)}\left(\boldsymbol{X}_n\boldsymbol{X}_n^T - (n-f(n)+1)\overline{\boldsymbol{X}}_n\overline{\boldsymbol{X}}_n^T\right) + \boldsymbol{S}_n
\end{aligned}
$$

If $f(n) = f(n-1) + 1$ then the latest observation replaces the oldest observation.

$$
\begin{aligned}
\overline{\boldsymbol{X}}_n &= \frac{1}{n - f(n) + 1} \sum_{i=f(n)}^{n} \boldsymbol{X}_i \\
&= \frac{1}{n - 1 - f(n-1) + 1} \sum_{i=f(n-1)}^{n-1} \boldsymbol{X}_i + \frac{1}{n - f(n) + 1} \left(\boldsymbol{X}_n - \boldsymbol{X}_{f(n-1)} \right) \\
&= \overline{\boldsymbol{X}}_{n-1} + \frac{1}{n - f(n) + 1} \left(\boldsymbol{X}_n - \boldsymbol{X}_{f(n)-1} \right) \\
\boldsymbol{\Sigma}_n &= \frac{w_n}{n - f(n)} \left(\sum_{i=f(n)}^{n} \boldsymbol{X}_i \boldsymbol{X}_i^T - (n - f(n) + 1) \overline{\boldsymbol{X}}_n \overline{\boldsymbol{X}}_n^T \right) + \boldsymbol{S}_n \\
&= \frac{w_n}{n - f(n)} \Bigg(\sum_{i=f(n-1)}^{n-1} \boldsymbol{X}_i \boldsymbol{X}_i^T + \boldsymbol{X}_n \boldsymbol{X}_n^T - \boldsymbol{X}_{f(n-1)} \boldsymbol{X}_{f(n-1)}^T \\
&\qquad - (n - f(n) + 1) \overline{\boldsymbol{X}}_n \overline{\boldsymbol{X}}_n^T \Bigg) + \boldsymbol{S}_n \\
&= \frac{w_n}{n - f(n)} \Bigg(\sum_{i=f(n-1)}^{n-1} \boldsymbol{X}_i \boldsymbol{X}_i^T - (n - f(n-1)) \overline{\boldsymbol{X}}_{n-1} \overline{\boldsymbol{X}}_{n-1}^T + (n - f(n-1)) \overline{\boldsymbol{X}}_{n-1} \overline{\boldsymbol{X}}_{n-1}^T \\
&\qquad - (n - f(n) + 1) \overline{\boldsymbol{X}}_n \overline{\boldsymbol{X}}_n^T + \boldsymbol{X}_n \boldsymbol{X}_n^T - \boldsymbol{X}_{f(n-1)} \boldsymbol{X}_{f(n-1)}^T \Bigg) + \boldsymbol{S}_n \\
&= w_n \mathrm{cov}(\boldsymbol{X}_{f(n-1)}, \ldots, \boldsymbol{X}_{n-1}) + \boldsymbol{S}_n \\
&\quad + \frac{w_n}{n - f(n)} \left((n - f(n) + 1) \left[\overline{\boldsymbol{X}}_{n-1} \overline{\boldsymbol{X}}_{n-1}^T - \overline{\boldsymbol{X}}_n \overline{\boldsymbol{X}}_n^T \right] + \boldsymbol{X}_n \boldsymbol{X}_n^T - \boldsymbol{X}_{f(n-1)} \boldsymbol{X}_{f(n-1)}^T \right) \\
&= \frac{w_n}{w_{n-1}} \left(\boldsymbol{\Sigma}_{n-1} - \boldsymbol{S}_{n-1} \right) + \boldsymbol{S}_n \\
&\quad + \frac{w_n}{n - f(n)} \left(\boldsymbol{X}_n \boldsymbol{X}_n^T - \boldsymbol{X}_{f(n)-1} \boldsymbol{X}_{f(n)-1}^T + (n - f(n) + 1) \left[\overline{\boldsymbol{X}}_{n-1} \overline{\boldsymbol{X}}_{n-1}^T - \overline{\boldsymbol{X}}_n \overline{\boldsymbol{X}}_n^T \right] \right)
\end{aligned}
$$

References

ANDRIEU, C. & THOMS, J. (2008). A tutorial on adaptive MCMC. *Statistics and Computing* **18**, 343–373.

ATCHADÉ, Y.F. & ROSENTHAL, J.S. (2005). On adaptive Markov chain Monte Carlo algorithms. *Bernoulli* **11**, 815–828.

CRAIU, R.V., GRAY, L., ŁATUSZYŃSKI, K., MADRAS, N., ROBERTS, G.O. & ROSENTHAL, J.S. (2015). Stability of adversarial Markov chains, with an application to adaptive MCMC algorithms. *The Annals of Applied Probability* **25**, 3592–3623.

GARTHWAITE, P.H., FAN, Y. & SISSON, S.A. (2016). Adaptive optimal scaling of Metropolis–Hastings algorithms using the Robbins–Monro process. *Communications in Statistics–Theory and Methods* **45**, 5098–5111.

GELMAN, A., ROBERTS, G.O. & GILKS, W.R. (1996). Efficient Metropolis jumping rules. *Bayesian Statistics* **5**, 42.

GEYER, C.J. (1991). Markov chain Monte Carlo maximum likelihood. *Comput Sci Stat* **23**, 156–163.

HAARIO, H., LAINE, M., MIRA, A. & SAKSMAN, E. (2006). DRAM: efficient adaptive MCMC. *Statistics and Computing* **16**, 339–354.

HAARIO, H., SAKSMAN, E. & TAMMINEN, J. (1999). Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics* **14**, 375–396.

HAARIO, H., SAKSMAN, E., TAMMINEN, J. et al. (2001). An adaptive Metropolis algorithm. *Bernoulli* **7**, 223–242.

HASTINGS, W. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**, 97–109.

HOLLINGSWORTH, T.D., ADAMS, E.R., ANDERSON, R.M., ATKINS, K., BARTSCH, S., BASÁÑEZ, M.G., BEHREND, M., BLOK, D.J., CHAPMAN, L.A., COFFENG, L. et al. (2015). Quantitative analyses and modelling to support achievement of the 2020 goals for nine neglected tropical diseases. *Parasites & Vectors* **8**, 630.

KEELING, M.J. & ROHANI, P. (2011). *Modeling infectious diseases in humans and animals*. Princeton University Press.

LEE, C., NEAL, P. et al. (2018). Optimal scaling of the independence sampler: Theory and practice. *Bernoulli* **24**, 1636–1652.

MARINARI, E. & PARISI, G. (1992). Simulated tempering: a new Monte Carlo scheme. *EPL (Europhysics Letters)* **19**, 451.

METROPOLIS, N., ROSENBLUTH, A.W., ROSENBLUTH, M.N., TELLER, A.H. & TELLER, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* **21**, 1087–1092.

MIASOJEDOW, B., MOULINES, E. & VIHOLA, M. (2013). An adaptive parallel tempering algorithm. *Journal of Computational and Graphical Statistics* **22**, 649–664.

NEAL, R.M. (2011). Mcmc using hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, pp. 139–188.

PLUMMER, M., BEST, N., COWLES, K. & VINES, K. (2006). Coda: Convergence diagnosis and output analysis for mcmc. *R News* **6**, 7–11. URL `https://journal.r-project.org/archive/`.

PRINCE, S.J. (2012). *Computer vision: models, learning, and inference*. Cambridge University Press.

ROBBINS, H. & MONRO, S. (1951). A stochastic approximation method. *The annals of Mathematical Statistics* , 400–407.

ROBERT, C.P., ELVIRA, V., TAWN, N. & WU, C. (2018). Accelerating MCMC algorithms. *Wiley Interdisciplinary Reviews: Computational Statistics* **10**, e1435.

ROBERTS, G.O., GELMAN, A. & GILKS, W.R. (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability* **7**, 110–120.

ROBERTS, G.O. & ROSENTHAL, J.S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science* **16**, 351–367.

ROBERTS, G.O. & ROSENTHAL, J.S. (2007). Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of Applied Probability* **44**, 458–475.

ROBERTS, G.O. & ROSENTHAL, J.S. (2009). Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics* **18**, 349–367.

ROBERTS, G.O. & TWEEDIE, R.L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli* **2**, 341–363.

SHERLOCK, C., FEARNHEAD, P. & ROBERTS, G.O. (2010). The random walk Metropolis: linking theory and practice through a case study. *Statistical Science* **25**, 172–190.

TAWN, N. (2017). Towards optimality of the parallel tempering algorithm. Ph.D. thesis, University of Warwick.

TAWN, N.G. & ROBERTS, G.O. (2019). Accelerating parallel tempering: Quantile tempering algorithm (QuanTA). *Advances in Applied Probability* **51**, 802–834.

TAWN, N.G., ROBERTS, G.O. & ROSENTHAL, J.S. (2020). Weight-preserving simulated tempering. *Statistics and Computing* **30**, 27–41.

VIHOLA, M. (2011). On the stability and ergodicity of adaptive scaling Metropolis algorithms. *Stochastic Processes and Their Applications* **121**, 2839–2860.

YANG, J. & ROSENTHAL, J.S. (2017). Automatically tuned general-purpose MCMC via new adaptive diagnostics. *Computational Statistics* **32**, 315–348.