Taylor & Francis
Taylor & Francis Group

Journal of Computational and Graphical Statistics

ISSN: 1061-8600 (Print) 1537-2715 (Online) Journal homepage: https://www.tandfonline.com/loi/ucgs20

A Repelling–Attracting Metropolis Algorithm for Multimodality

Hyungsuk Tak, Xiao-Li Meng & David A. van Dyk

To cite this article: Hyungsuk Tak, Xiao-Li Meng & David A. van Dyk (2018) A Repelling–Attracting Metropolis Algorithm for Multimodality, Journal of Computational and Graphical Statistics, 27:3, 479-490, DOI: 10.1080/10618600.2017.1415911

To link to this article: https://doi.org/10.1080/10618600.2017.1415911

JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS
2018, VOL. 27, NO. 3, 479–490
https://doi.org/10.1080/10618600.2017.1415911

Taylor & Francis
Taylor & Francis Group

A Repelling–Attracting Metropolis Algorithm for Multimodality

Hyungsuk Tak [a], Xiao-Li Meng [b], and David A. van Dyk [c]

[a]Statistical and Applied Mathematical Sciences Institute, Durham, NC; [b]Department of Statistics, Harvard University, Cambridge, MA; [c]Statistics Section, Department of Mathematics, Imperial College London, London, UK

ABSTRACT

Although the Metropolis algorithm is simple to implement, it often has difficulties exploring multimodal distributions. We propose the repelling–attracting Metropolis (RAM) algorithm that maintains the simple-to-implement nature of the Metropolis algorithm, but is more likely to jump between modes. The RAM algorithm is a Metropolis-Hastings algorithm with a proposal that consists of a downhill move in density that aims to make local modes repelling, followed by an uphill move in density that aims to make local modes attracting. The downhill move is achieved via a reciprocal Metropolis ratio so that the algorithm prefers downward movement. The uphill move does the opposite using the standard Metropolis ratio which prefers upward movement. This down-up movement in density increases the probability of a proposed move to a different mode. Because the acceptance probability of the proposal involves a ratio of intractable integrals, we introduce an auxiliary variable which creates a term in the acceptance probability that cancels with the intractable ratio. Using several examples, we demonstrate the potential for the RAM algorithm to explore a multimodal distribution more efficiently than a Metropolis algorithm and with less tuning than is commonly required by tempering-based methods. Supplementary materials are available online.

ARTICLE HISTORY
Received September 2016
Revised October 2017

KEYWORDS
Auxiliary variable; Equi-energy sampler; Forced Metropolis transition; Markov chain Monte Carlo; Parallel tempering; Tempered transitions

1. Introduction and Overview

Multimodal distributions are common in statistical applications. However, the Metropolis algorithm (Metropolis et al. 1953), one of the most widely used Markov chain Monte Carlo (MCMC) methods, tends to produce Markov chains that do not readily jump between local modes. A popular MCMC strategy for dealing with multimodality is tempering such as parallel tempering (Geyer 1991), simulated tempering (Geyer and Thompson 1995), tempered transitions (Neal 1996), and equi-energy sampler (Kou, Zhou, and Wong 2006). Though powerful, these methods typically require extensive tuning.

Building on Metropolis, we construct an alternative multimodal sampler called the repelling-attracting Metropolis (RAM) algorithm, which is essentially as easy to implement as the original Metropolis algorithm. RAM encourages a Markov chain to jump between modes more frequently than Metropolis, and with less tuning requirements than tempering methods. Since RAM is more likely to jump between modes than Metropolis, the proportions of its iterations that are associated with each mode are more reliable estimates of their relative masses.

RAM generates a proposal via forced downhill and forced uphill Metropolis transitions. The term *forced* emphasizes that neither Metropolis transition is allowed to stay at its current state because we repeatedly make proposals until one is accepted. The forced downhill Metropolis transition uses a reciprocal ratio of the target densities in its acceptance probability. This encourages the intermediate proposal to prefer downward moves since a lower density state has a higher chance of being accepted, hence local modes become *repelling*. The subsequent forced uphill Metropolis transition generates a final proposal with a standard Metropolis ratio that makes local modes *attracting*. Together the downhill and uphill transitions form a proposal for a Metropolis-Hastings (MH) sampler (Hastings 1970), as shown in Figure 1; a final accept–reject step preserves the stationary distribution.

As with other MH samplers, the normalizing constant of the target density need not be known, but the scale of the (symmetric) jumping rules used within the downhill and uphill transitions needs to be tuned. In principle, RAM is designed to improve Metropolis' ability to jump between modes using the same jumping rule as Metropolis where this jumping rule is tuned to optimize the underlying Metropolis sampler for the multimodal target. One could do still better with additional tuning of RAM, but in our experience even with no additional tuning, RAM can perform better than its underlying Metropolis sampler.

Although we can draw a sample using the down-up jumping rule, the overall acceptance probability contains a ratio of intractable integrals. We can avoid evaluating this ratio by introducing an auxiliary variable (Møller et al. 2006). This preserves the target marginal distribution and requires another forced downhill Metropolis transition for the auxiliary variable. Thus, RAM generates a proposal via three forced Metropolis transitions but accepts the proposal with an easy-to-compute acceptance probability.

RAM is related to a number of existing algorithms. The down-up proposal of RAM may be viewed as a simpler version

CONTACT Hyungsuk Tak *hyungsuk.tak@gmail.com* Statistical and Applied Mathematical Sciences Institute.
Supplementary materials for this article are available online. Please go to *www.tandfonline.com/r/JCGS*.

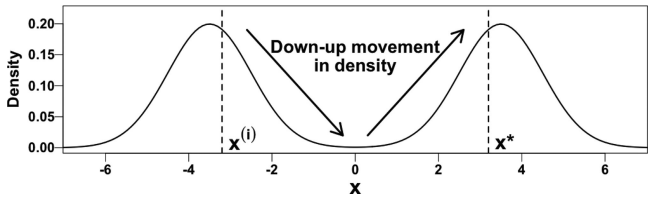

Figure 1. A repelling-attracting Metropolis algorithm is a Metropolis-Hastings algorithm that generates a proposal x^* given the current state $x^{(i)}$ by making a down-up movement in density, that is, repelling–attracting to local modes, via forced downhill and uphill Metropolis transitions. The proposal x^* has a higher chance to be near a mode other than the one of the current state, and it is then accepted or rejected in the usual way to preserve the stationary distribution.

of a mode-jumping proposal (Tjelmeland and Hegstad 2001), whose uphill movement is achieved by a deterministic optimizer. Also, the forced Metropolis transition of RAM is similar to the delayed rejection method (Tierney and Mira 1999; Trias, Vecchio, and Veitch 2009) in that both generate proposals repeatedly until one is accepted. RAM's forced transition is a special case of the delayed rejection method in that RAM uses the same jumping rule throughout, while delayed rejection allows different jumping rules.

In a series of four numerical examples, we compare RAM's performance to Metropolis and commonly used tempering-based methods such as the equi-energy sampler, parallel tempering, and tempered transitions. We adjust for the required number of evaluations of the target density or the overall CPU time required by each sampler. Our examples range from relatively simple and high dimensional Gaussian mixtures (Examples 1 and 2) to lower dimensional, but more complex targets that arise as posterior distributions in scientific problems (Examples 3 and 4). We compare RAM with standard Metropolis, implementing both samplers with a common jumping rule that is tuned to improve the mixing of Metropolis for the particular multimodal target distribution. These comparisons suggest that replacing Metropolis with RAM when targeting a multimodal distribution can be an efficient strategy, in terms of user's effort.

In our comparisons with tempering-based samplers, we find that in moderate dimensions RAM performs as well as or better than tempering-based methods, without the subtle tuning that these methods require. Even with a higher dimensional target distribution in Example 3, we show how RAM can be embedded within a Gibbs sampler to obtain results as good as tempering-based methods, again without the tuning they require. Because RAM is able to jump between modes relatively often, it provides good estimates of the relative size of the modes. In our examples, RAM obtains more reliable estimates of the mode sizes than Metropolis and is easier to directly implement than tempering-based methods.

2. A Repelling–Attracting Metropolis Algorithm

2.1. A Down-up Proposal

We briefly review MH. A transition kernel on $\mathbf{R}^d$, denoted by $P(B \mid x)$, is the conditional probability distribution of a transition from $x \in \mathbf{R}^d$ to a point in a Borel set B in $\mathbf{R}^d$. Hence, $P(\mathbf{R}^d \mid x) = 1$ and $P(\{x\} \mid x)$ need not be zero (Chib and Greenberg 1995). A jumping density given the current state $x^{(i)}$ is the conditional density with respect to Lebesgue measure that generates a proposal x^*, denoted by $q(x^* \mid x^{(i)})$. With a target density denoted by π, either normalized or unnormalized, the transition kernel of MH is

$$P(dx^* \mid x^{(i)}) = q(x^* \mid x^{(i)})\alpha(x^* \mid x^{(i)})dx^* + \delta_{x^{(i)}}(dx^*)\{1 - A(x^{(i)})\}, \quad (1)$$

where the Dirac measure $\delta_{x^{(i)}}(dx^*)$ is one if $x^{(i)} \in dx^*$ and zero otherwise and $\alpha(x^* \mid x^{(i)})$ is the probability of accepting the proposal and setting $x^{(i+1)} = x^*$, that is,

$$\alpha(x^* \mid x^{(i)}) = \min\left\{1, \frac{\pi(x^*)q(x^{(i)} \mid x^*)}{\pi(x^{(i)})q(x^* \mid x^{(i)})}\right\}.$$

Here, $1 - A(x^{(i)})$ is the probability of staying at $x^{(i)}$, that is, of setting $x^{(i+1)} = x^{(i)}$, and thus $A(x^{(i)})$ is the probability of moving away from $x^{(i)}$:

$$A(x^{(i)}) = \int q(x^* \mid x^{(i)})\alpha(x^* \mid x^{(i)})dx^*.$$

If the jumping density is symmetric, that is, $q(a \mid b) = q(b \mid a)$, MH reduces to Metropolis with

$$\alpha(x^* \mid x^{(i)}) = \min\left\{1, \frac{\pi(x^*)}{\pi(x^{(i)})}\right\}. \quad (2)$$

We assume that q is symmetric hereafter because RAM is currently feasible only with a symmetric q, that is, RAM can replace any Metropolis but not the more general MH algorithm.

Metropolis is one of the most commonly used MCMC methods, but it often has difficulties exploring multimodal distributions. Alternative tempering methods usually require more tuning, which can be restrictive to practitioners. RAM maintains the simple-to-implement nature of Metropolis, but is more likely to jump between modes. The key to RAM is a down-up jumping density that generates a proposal x^* after making a down-up movement in density. Because the corresponding acceptance probability is intractable, we generate an auxiliary variable z^* given x^* in such a way that the acceptance probability becomes computable. Thus, RAM is an MH algorithm with a unique joint jumping density $q^{\mathrm{DU}}(x^* \mid x^{(i)})q^{\mathrm{D}}(z^* \mid x^*)$ and an easy-to-compute acceptance probability $\alpha^{\mathrm{J}}(x^*, z^* \mid x^{(i)}, z^{(i)})$ that preserves the target marginal distribution $\pi(x)$. Next, we describe q^{DU}, q^{D}, and α^{J}.

The down-up jumping density, $q^{\mathrm{DU}}(x^* \mid x^{(i)})$, first generates an intermediate downhill proposal x' given the current state $x^{(i)}$ and then an uphill proposal x^* given x', that is,

$$q^{\mathrm{DU}}(x^* \mid x^{(i)}) = \int q^{\mathrm{D}}(x' \mid x^{(i)})q^{\mathrm{U}}(x^* \mid x')dx',$$

where q^{D} and q^{U} can be any conditional density functions that prefer lower and higher density states than the given states, respectively. Our choice for q^{D} is a forced downhill Metropolis kernel density defined as

$$q^{\mathrm{D}}(x' \mid x^{(i)}) = \frac{q(x' \mid x^{(i)})\alpha_\epsilon^{\mathrm{D}}(x' \mid x^{(i)})}{A^{\mathrm{D}}(x^{(i)})}, \quad (3)$$

where

$$\alpha_\epsilon^{\mathrm{D}}(x' \mid x^{(i)}) = \min\left\{1, \frac{\pi(x^{(i)}) + \epsilon}{\pi(x') + \epsilon}\right\} \quad (4)$$

is the probability of accepting an intermediate proposal x' drawn from $q(x' \mid x^{(i)})$ and $A^{\mathrm{D}}(x^{(i)}) = \int q(x' \mid x^{(i)})\alpha_\epsilon^{\mathrm{D}}(x' \mid x^{(i)})dx'$ is the normalizing constant[1]. We use the term *forced* because this Metropolis transition kernel repeatedly generates intermediate proposals (like rejection sampling) until one is accepted. Also, we use the term *downhill* because the reciprocal of the ratio of the target densities in (4) makes local modes repelling rather than attracting: If the density of x' is smaller than that of $x^{(i)}$, x' is accepted with probability one. The appearance of ϵ in $\alpha_\epsilon^{\mathrm{D}}(x' \mid x^{(i)})$ is discussed below.

Similarly, we set q^{U} to a forced uphill Metropolis transition kernel density defined as

$$q^{\mathrm{U}}(x^* \mid x') = \frac{q(x^* \mid x')\alpha_\epsilon^{\mathrm{U}}(x^* \mid x')}{A^{\mathrm{U}}(x')},$$

where

$$\alpha_\epsilon^{\mathrm{U}}(x^* \mid x') = \min\left\{1, \frac{\pi(x^*) + \epsilon}{\pi(x') + \epsilon}\right\} \tag{5}$$

is the probability of accepting a proposal x^* generated from $q(x^* \mid x')$ and $A^{\mathrm{U}}(x') = \int q(x^* \mid x')\alpha_\epsilon^{\mathrm{U}}(x^* \mid x')dx^*$ is the normalizing constant. This kernel restores the attractiveness of local modes because $\alpha_\epsilon^{\mathrm{U}}(x^* \mid x')$ is a typical Metropolis acceptance probability except that ϵ is added for numerical stability; both $\pi(x')$ and $\pi(x^*)$ can be nearly zero when both x' and x^* are in a valley between modes. The value of ϵ may affect the convergence rate. To minimize its impact on the acceptance probability in (5), we choose ϵ to be small with a default choice of $\epsilon = 10^{-308}$, the smallest power of ten that R (R Core Team 2016) treats as positive. For symmetry, we use ϵ in the same way in the acceptance probability of the downhill transition in (4). Consequently, our choices for q^{D} and q^{U} satisfy $\int q^{\mathrm{DU}}(x^* \mid x^{(i)})dx^* = 1$.

Without forced transitions, the final proposal x^* could be the same as the current state $x^{(i)}$ after consecutive rejections in both the downhill and uphill Metropolis transitions, or x^* could be generated via only one of the downhill and uphill transitions if the other were rejected. This would not be helpful for our purposes because it would not induce a down-up movement. Moreover, a forced transition kernel is mathematically simpler than that of Metropolis in that it eliminates the term, $\delta_{x^{(i)}}(dx^*)\{1 - A(x^{(i)})\}$ in (1).

The MH acceptance probability with the down-up jumping density q^{DU} simplifies to

$$\begin{aligned}\alpha^{\mathrm{DU}}(x^* \mid x^{(i)}) &= \min\left\{1, \frac{\pi(x^*)q^{\mathrm{DU}}(x^{(i)} \mid x^*)}{\pi(x^{(i)})q^{\mathrm{DU}}(x^* \mid x^{(i)})}\right\} \\ &= \min\left\{1, \frac{\pi(x^*)A^{\mathrm{D}}(x^{(i)})}{\pi(x^{(i)})A^{\mathrm{D}}(x^*)}\right\},\end{aligned} \tag{6}$$

where the last equality holds because

$$\begin{aligned}&q^{\mathrm{DU}}(x^* \mid x^{(i)})A^{\mathrm{D}}(x^{(i)}) \\ &= \int q(x' \mid x^{(i)})\alpha_\epsilon^{\mathrm{D}}(x' \mid x^{(i)})\frac{q(x^* \mid x')\alpha_\epsilon^{\mathrm{U}}(x^* \mid x')}{A^{\mathrm{U}}(x')}dx' \\ &= \int q(x^{(i)} \mid x')\alpha_\epsilon^{\mathrm{U}}(x^{(i)} \mid x')\frac{q(x' \mid x^*)\alpha_\epsilon^{\mathrm{D}}(x' \mid x^*)}{A^{\mathrm{U}}(x')}dx' \\ &= q^{\mathrm{DU}}(x^{(i)} \mid x^*)A^{\mathrm{D}}(x^*),\end{aligned}$$

and thus

$$\frac{q^{\mathrm{DU}}(x^{(i)} \mid x^*)}{q^{\mathrm{DU}}(x^* \mid x^{(i)})} = \frac{A^{\mathrm{D}}(x^{(i)})}{A^{\mathrm{D}}(x^*)}. \tag{7}$$

2.2. An Auxiliary Variable Approach

Since the ratio of the normalizing constants in (7) is intractable, we use an auxiliary variable approach (Møller et al. 2006) to avoid its evaluation in (6). We form a joint Markov chain for x and an auxiliary variable z so that the target marginal density for x is still π, yet the resulting joint MH algorithm has an easily computable acceptance ratio. Specifically, after generating x^* via q^{DU}, we generate z^* given x^* using the forced downhill Metropolis kernel density q^{D} in (3), which typically requires one evaluation of the target density on average. We set the joint target density $\pi(x, z) = \pi(x)q(z \mid x)$, which then leads to, as we shall prove shortly, the acceptance probability of the joint jumping density $q^{\mathrm{DU}}(x^* \mid x^{(i)})q^{\mathrm{D}}(z^* \mid x^*)$ as

$$\alpha^{\mathrm{J}}(x^*, z^* \mid x^{(i)}, z^{(i)}) = \min\left\{1, \frac{\pi(x^*)\min\{1, \frac{\pi(x^{(i)})+\epsilon}{\pi(z^{(i)})+\epsilon}\}}{\pi(x^{(i)})\min\{1, \frac{\pi(x^*)+\epsilon}{\pi(z^*)+\epsilon}\}}\right\}. \tag{8}$$

Consequently, introducing z results in the easy-to-compute acceptance probability in (8). RAM accepts the joint proposal (x^*, z^*) as $(x^{(i+1)}, z^{(i+1)})$ with the probability in (8) and sets $(x^{(i+1)}, z^{(i+1)})$ to $(x^{(i)}, z^{(i)})$ otherwise. Since RAM is an MH algorithm, it automatically satisfies the detailed balance condition. We notice that in (8), $\pi(z^{(i)})$ is likely to be smaller than $\pi(x^{(i)})$ because $z^{(i)}$ is generated by the forced downhill transition. Similarly, $\pi(z^*)$ is likely to be smaller than $\pi(x^*)$. When $z^{(i)}$ and z^* have lower target densities than $x^{(i)}$ and x^*, respectively (likely, but not required), the acceptance probability in (8) reduces to the acceptance probability of Metropolis in (2).

We obtained (8) by considering a joint target distribution $\pi(x, z) = \pi(x)\pi^{\mathrm{C}}(z \mid x)$, with a joint jumping density in the form of

$$\begin{aligned}q^{\mathrm{J}}(x^*, z^* \mid x^{(i)}, z^{(i)}) &= q_1(x^* \mid x^{(i)}, z^{(i)})q_2(z^* \mid x^*, x^{(i)}, z^{(i)}) \\ &= q_1(x^* \mid x^{(i)})q_2(z^* \mid x^*).\end{aligned} \tag{9}$$

The MH acceptance probability for the joint proposal then is

$$\begin{aligned}&\alpha^{\mathrm{J}}(x^*, z^* \mid x^{(i)}, z^{(i)}) \\ &= \min\left\{1, \frac{\pi(x^*)\pi^{\mathrm{C}}(z^* \mid x^*)q_1(x^{(i)} \mid x^*)q_2(z^{(i)} \mid x^{(i)})}{\pi(x^{(i)})\pi^{\mathrm{C}}(z^{(i)} \mid x^{(i)})q_1(x^* \mid x^{(i)})q_2(z^* \mid x^*)}\right\},\end{aligned} \tag{10}$$

which recalls the pseudo-marginal approach (Beaumont 2003; Andrieu and Roberts 2009) that uses an unbiased estimator of an intractable target density. In this setting, however, it is the jumping density that is intractable. Somewhat surprisingly, there does not seem to be an easy way to modify the pseudo-marginal argument, other than directly following the more general auxiliary variable approach in Møller et al. (2006).

Specifically, suppose we are able to sample from q_1 in (9) but are not able to evaluate q_1. We can find a function f such that

[1] This normalizing constant $A^{\mathrm{D}}(x^{(i)})$ is finite if q is a proper density, that is, $\int q(x' \mid x^{(i)})dx' < \infty$, because $\alpha_\epsilon^{\mathrm{D}}(x' \mid x^{(i)})$ is bounded between 0 and 1. Similarly, $A^{\mathrm{U}}(x')$ appearing later is also finite if q is proper.

$q_1(x^{(i)} \mid x^*)/q_1(x^* \mid x^{(i)}) = f(x^{(i)})/f(x^*)$ because the ratio of two (compatible) conditional densities equals the corresponding ratio of marginal densities, where f itself may or may not be computable. If we can find a function q_2 in (9) whose normalizing constant is proportional to f, then the joint acceptance probability in (10) becomes free of the intractable quantities.

For RAM, we set $q_1(x^* \mid x^{(i)}) = q^{\mathrm{DU}}(x^* \mid x^{(i)})$, and thus $f(x^{(i)}) = A^{\mathrm{D}}(x^{(i)})$. To eliminate this intractable normalizing constant, we choose $q_2(z^* \mid x^*) = q^{\mathrm{D}}(z^* \mid x^*)$. Since Møller et al. (2006) suggested choosing π^{C} similar to q_2, we choose $\pi^{\mathrm{C}}(z^* \mid x^*) = q(z^* \mid x^*)$. With these choices, the acceptance probability in (10) reduces to (8) because

$$
\begin{aligned}
&\alpha^{\mathrm{J}}(x^*, z^* \mid x^{(i)}, z^{(i)}) \\
&= \min\left\{1, \frac{\pi(x^*)q(z^* \mid x^*)q^{\mathrm{DU}}(x^{(i)} \mid x^*)q^{\mathrm{D}}(z^{(i)} \mid x^{(i)})}{\pi(x^{(i)})q(z^{(i)} \mid x^{(i)})q^{\mathrm{DU}}(x^* \mid x^{(i)})q^{\mathrm{D}}(z^* \mid x^*)}\right\} \\
&= \min\left\{1, \frac{\pi(x^*)q(z^* \mid x^*)A^{\mathrm{D}}(x^{(i)})q(z^{(i)} \mid x^{(i)})\alpha_\epsilon^{\mathrm{D}}(z^{(i)} \mid x^{(i)})/A^{\mathrm{D}}(x^{(i)})}{\pi(x^{(i)})q(z^{(i)} \mid x^{(i)})A^{\mathrm{D}}(x^*)q(z^* \mid x^*)\alpha_\epsilon^{\mathrm{D}}(z^* \mid x^*)/A^{\mathrm{D}}(x^*)}\right\} \\
&= \min\left\{1, \frac{\pi(x^*)\alpha_\epsilon^{\mathrm{D}}(z^{(i)} \mid x^{(i)})}{\pi(x^{(i)})\alpha_\epsilon^{\mathrm{D}}(z^* \mid x^*)}\right\} \\
&= \min\left\{1, \frac{\pi(x^*)\min\{1, \frac{\pi(x^{(i)})+\epsilon}{\pi(z^{(i)})+\epsilon}\}}{\pi(x^{(i)})\min\{1, \frac{\pi(x^*)+\epsilon}{\pi(z^*)+\epsilon}\}}\right\},
\end{aligned}
$$

where the second equality follows from (3) and (7), and the last equality follows from (4).

2.3. Implementation of the RAM Algorithm

Each RAM iteration is composed of the four steps in Table 1. The first three generate a joint proposal, (x^*, z^*), via three consecutive forced transitions; Step 1 is the downward proposal x' given $x^{(i)}$, Step 2 is the upward proposal x^* given x', and Step 3 is the downward proposal z^* given x^*. Finally, Step 4 determines if the joint proposal is accepted. In our numerical examples, the downhill proposals in Steps 1 and 3 are usually accepted on the first try. However, the number of proposals needed for the uphill move in Step 2 varies. As the dimension increases, for instance, generating a higher density proposal becomes challenging, and the uphill transition in Step 2 requires more proposals.

Some density values used by RAM do not need to be calculated repeatedly. For example, since the density of the previous value $\pi(x^{(i)})$ is used in both Steps 1 and 4, it is better to evaluate and cache this value before Step 1. Also, $\pi(x')$ in Step 2 is evaluated during the final forced downhill step in Step 1, and can be cached and reused in Step 2. Similarly, we can cache the values of $\pi(x^*)$ and $\pi(z^*)$ in Steps 2 and 3, respectively. The cached values can also be used to compute the acceptance probability in Step 4. In our numerical illustrations, we use an equivalent caching policy for other algorithms. For example, an MH transition can be efficiently coded to evaluate the target density once at each iteration (only for the density of a proposal) by caching the density of the current state.

Table 1. A repelling–attracting Metropolis algorithm.

Set initial values $x^{(0)}$ and $z^{(0)}$ $(= x^{(0)})$. For $i = 0, 1, \ldots$

Step 1: ($\searrow$) Repeatedly sample $x' \sim q(x' \mid x^{(i)})$ and $u_1 \sim \mathrm{Uniform}(0,1)$ until $u_1 < \min\left\{1, \frac{\pi(x^{(i)})+\epsilon}{\pi(x')+\epsilon}\right\}$.

Step 2: ($\nearrow$) Repeatedly sample $x^* \sim q(x^* \mid x')$ and $u_2 \sim \mathrm{Uniform}(0,1)$ until $u_2 < \min\left\{1, \frac{\pi(x^*)+\epsilon}{\pi(x')+\epsilon}\right\}$.

Step 3: ($\searrow$) Repeatedly sample $z^* \sim q(z^* \mid x^*)$ and $u_3 \sim \mathrm{Uniform}(0,1)$ until $u_3 < \min\left\{1, \frac{\pi(x^*)+\epsilon}{\pi(z^*)+\epsilon}\right\}$.

Step 4: Set $(x^{(i+1)}, z^{(i+1)}) = (x^*, z^*)$ if $u_4 < \min\left\{1, \frac{\pi(x^*)\min\{1,(\pi(x^{(i)})+\epsilon)/(\pi(z^{(i)})+\epsilon)\}}{\pi(x^{(i)})\min\{1,(\pi(x^*)+\epsilon)/(\pi(z^*)+\epsilon)\}}\right\}$, where $u_4 \sim \mathrm{Uniform}(0,1)$, and set $(x^{(i+1)}, z^{(i+1)}) = (x^{(i)}, z^{(i)})$ otherwise.

RAM can replace a Metropolis kernel within a Gibbs sampler. Suppose we have a Gibbs sampler that iteratively samples $\pi_1(x \mid y)$ and $\pi_2(y \mid x)$, and a Metropolis kernel that is invariant to $\pi_1(x \mid y)$ is used within the Gibbs sampler. To replace Metropolis with RAM, we keep track of the auxiliary variable z during the run. For example, once we sample $x^{(i)}$ and $z^{(i)}$ at iteration i via a RAM kernel that is (marginally) invariant to $\pi_1(x \mid y^{(i-1)})$, only $x^{(i)}$ is used to sample $\pi_2(y \mid x^{(i)})$, but $z^{(i)}$ is used to sample $x^{(i+1)}$ in the next iteration.

For simplicity, we use Gaussian jumping rules, though any symmetric density can be used. Specifically, we consider a d-dimensional Gaussian density with covariance matrix Σ as q in Table 1; both RAM and Metropolis share the same tuning parameter Σ. RAM is designed to improve the ability of Metropolis to jump between modes using a jumping rule that is tuned to optimize Metropolis for the multimodal target. In practice, this means a large jumping scale for unknown mode locations or a properly adjusted jumping scale for known mode locations. One could do still better with additional tuning of RAM. For example, if Σ is tuned to optimize Metropolis for a multimodal target, we can simply set the covariance matrix of q for RAM to $\Sigma/2$ because RAM's down-up proposal is generated by two (down-up) Metropolis transitions. In our numerical illustrations, we show that RAM can improve on Metropolis even without additional tuning. We introduce several useful strategies for tuning Σ, but their effectiveness may vary in different settings.

3. Numerical Illustrations

3.1. Example 1: A Mixture of Twenty Bivariate Gaussian Densities

To compare RAM with tempering methods, our first numerical illustration targets a mixture of 20 bivariate Gaussian distributions given by Kou, Zhou, and Wong (2006):

$$\pi(x) \propto \sum_{j=1}^{20} \frac{w_j}{\tau_j^2} \exp\left(-\frac{1}{2\tau_j^2}(x-\mu_j)^\top(x-\mu_j)\right),$$

where $x = (x_1, x_2)^\top$. The 20 mean vectors, $\{\mu_1, \ldots, \mu_{20}\}$, are specified in Kou, Zhou, and Wong (2006) and plotted in the first panel of Figure 2. Following Kou, Zhou, and Wong (2006), we consider two cases; in case (a), the modes are equally weighted and have equal variances, $w_j = 1/20$ and $\tau_j^2 = 1/100$, and in case (b) both weights and variances are unequal, that is, $w_j = 1/\|\mu_j - (5,5)^\top\|$ and $\tau_j^2 = \|\mu_j - (5,5)^\top\|/20$. In case (b), modes near (5, 5) have higher weight and smaller variances. Contour plots of the target distributions in cases (a) and (b) appear in Figure 2.

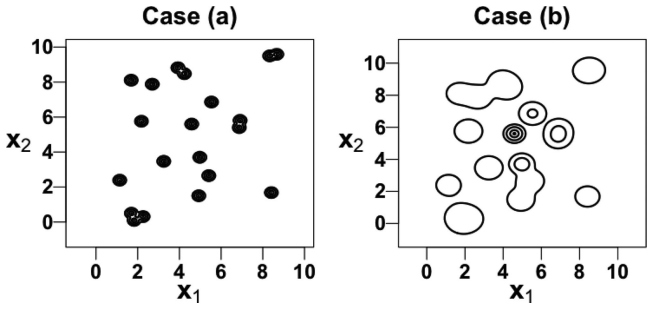

Figure 2. The first panel exhibits the contour plot of the target density in Example 1, case (a) and the second panel shows that of the target density in Example 1, case (b). The plotted contours outline regions with probability 1%, 10%, 50%, and 95% under $\pi(x)$.

Kou, Zhou, and Wong (2006) used this target distribution to compare the equi-energy sampler (EE) and parallel tempering (PT). We follow their simulation configurations by running RAM for 75,000 iterations for both cases, initializing the chain at random values of $x^{(0)}$ and $z^{(0)}$ in the unit square. Although Kou, Zhou, and Wong (2006) did not specify the burn-in size, we discard the first 25,000 iterations because they consistently use one-third of the iterations as burn-in in the other examples. We set q to be Gaussian with covariance matrix $\sigma^2 I_2$, where I_2 is the identity matrix. To tune σ, we initialize ten independent chains with 10 different values of $\sigma \in \{3.0, 3.5, \ldots, 7.5\}$. Following Kou, Zhou, and Wong (2006), we set σ to the value that leads to the best autocorrelation function among those that visit all modes. This is 4.0 in case (a) and 3.5 in case (b). The acceptance rate is 0.048 for case (a) and 0.228 for case (b).

Figure 3 gives bivariate scatterplots of the Monte Carlo sample of size 50,000 obtained with RAM for the two cases, bivariate trace plots of the last 2000 iterations for case (a) and the last 1000 iterations for case (b), and autocorrelation plots for x_1. Figure 3 can be compared to Figure 3 and Figure 4 of Kou, Zhou, and Wong (2006), which summarize the performance of EE and PT for cases (a) and (b), respectively.

To compare the accuracy of the moment estimates obtained with the algorithms, we again follow Kou, Zhou, and Wong (2006) and run 20 independent chains using RAM. Table 2 summarizes the comparisons, where the ratios of the mean squared error (MSE) of both EE and PT to that of RAM are all greater than one. The improvement is particularly striking for case (b). These indicate that RAM leads to a more reliable proportion of iterations that are associated with each mode across the 20 runs.

Finally, we compare the average evaluation cost of each algorithm by reporting the expected total number of evaluations of the target density π needed to obtain the final sample, including burn-in, divided by the final sample size; we denote this quantity by N_π^{X}, where "X" specifies the algorithm. As detailed in Appendix A, $N_\pi^{\mathrm{EE}} = 16.0$ and $N_\pi^{\mathrm{PT}} = 5.8$. For RAM, $N_\pi^{\mathrm{RAM}} = 7.1$ in case (a) and $N_\pi^{\mathrm{RAM}} = 5.0$ in case (b)[2]. More evaluations are needed for case (a) because the area of near zero density is much larger than that in case (b), see Figure 2, and a forced uphill transition thus requires more proposals (and thus more evaluations). Nonetheless, the number of target density evaluations (and thus CPU time) required by RAM indicates that the gain of using RAM in terms of MSE is competitive.

[2] The average number of proposals required by the forced downhill transition is 1.01 in case (a) and 1.06 in case (b), that of the uphill proposals is 4.70 in case (a) and 2.57 in case (b), and that of the downhill auxiliary variables is 1.39 in case (a) and 1.35 in case (b).

3.2. Example 2: High-Dimensional Multimodal Distributions

Consider an equal mixture of eight d-dimensional Gaussian distributions:

$$\pi(x) \propto \sum_{j=1}^{8} \exp\left(-\frac{1}{2}(x-\mu_j)^\top (x-\mu_j)\right), \tag{11}$$

where $x = (x_1, x_2, \ldots, x_d)^\top$ and the eight mean vectors are defined by setting their first three coordinates to the eight vertices of a cube of edge length 10 situated with its corner at the origin and their remaining coordinates are filled with (10, 0) or (0, 10) repeatedly:

$$\begin{aligned}
\mu_1 &= (10, 10, 10, 0, 10, 0, 10, \ldots, 0, 10),\\
\mu_2 &= (0, 0, 0, 10, 0, 10, 0, \ldots, 10, 0),\\
\mu_3 &= (10, 0, 10, 0, 10, 0, 10, \ldots, 0, 10),\\
\mu_4 &= (0, 10, 10, 0, 10, 0, 10, \ldots, 0, 10),\\
\mu_5 &= (0, 0, 10, 0, 10, 0, 10, \ldots, 0, 10),\\
\mu_6 &= (0, 10, 0, 10, 0, 10, 0, \ldots, 10, 0),\\
\mu_7 &= (10, 0, 0, 10, 0, 10, 0, \ldots, 10, 0),\\
\mu_8 &= (10, 10, 0, 10, 0, 10, 0, \ldots, 10, 0).
\end{aligned}$$

Suppose that the first two modes, μ_1 and μ_2, are known, perhaps from an initial search, while the other six modes are unknown. Here, we investigate RAM's ability to explore a high-dimensional distribution by using it to sample (11) with the five values of $d \in \{3, 5, 7, 9, 11\}$. We also compare RAM to both Metropolis and PT, taking into account their average evaluation cost, $\mathrm{N}_\pi^{\mathrm{X}}$, as defined in Section 3.1

We set q to be a d-dimensional Gaussian density with covariance matrix Σ. To achieve a reasonable acceptance rate, we first run two Metropolis chains each of length 5000, initialized at the two known mode locations and using a Gaussian jumping rule with covariance matrix $(2.38^2/d) \times I_d$, where I_d is the identity matrix. We then set Σ to the sample covariance matrix of the combined sample from the two chains. To improve Metropolis' ability to jump between modes, we reset Σ to the sample covariance matrix of the burn-in sample. This one-time adaptation does not affect the validity of the resulting chain.

For each d, we run RAM ten times to obtain ten chains each of length 500,000, discarding the first 200,000 iterations of each chain as burn-in. RAM's average evaluation cost $\mathrm{N}_\pi^{\mathrm{RAM}}$ is 6.54 for $d = 3$, 7.54 for $d = 5$, 8.45 for $d = 7$, 9.58 for $d = 9$, and 10.77 for $d = 11$. As d increases, RAM requires more evaluations because it is more difficult to find a proposal that increases the density in the forced uphill transition.

For each d, we also obtain ten chains each using both Metropolis and PT with the same Gaussian jumping rule used by RAM. PT runs five parallel chains under five temperature levels, 2^k for $k = 0, 1, \ldots, 4$, each of which uses Metropolis transitions. PT always proposes a single swap between a randomly

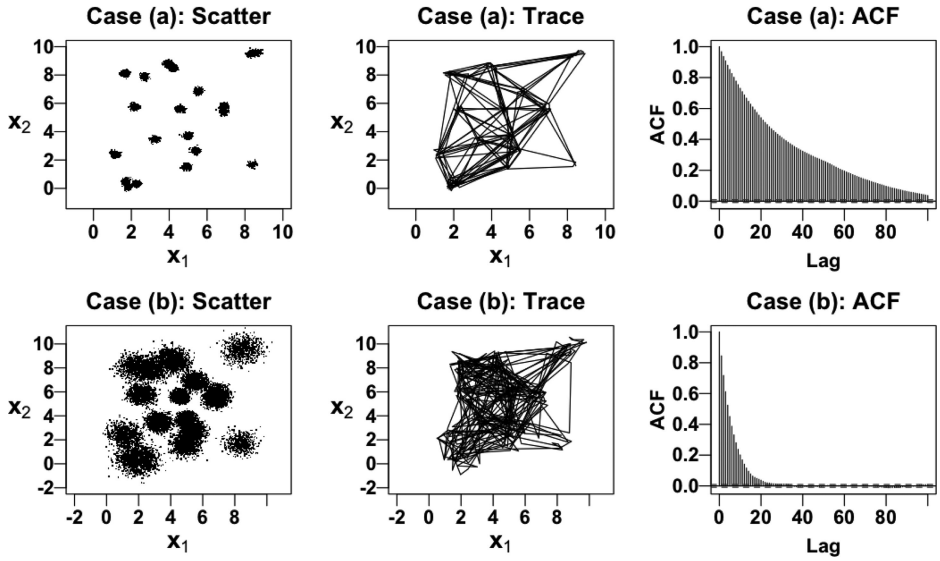

Figure 3. Results of the RAM algorithm. The first column displays bivariate scatterplots for 50,000 samples, the middle column displays the bivariate trace plots for the last 2000 samples for case (a) and the last 1000 samples for case (b), and the last column displays the autocorrelation functions for 50,000 samples of x_1.

Table 2. Moment estimates for cases (a) and (b) based on 20 independent chains, each of length 50,000, generated with RAM, EE (equi-energy sampler), and PT (parallel tempering). The results for EE and PT are from Kou, Zhou, and Wong (2006), and presented in their original format: Sample average (sample standard deviation) over the 20 replications.

Case (a)	Truth	RAM	EE	PT	MSE ratio (EE/RAM)	MSE ratio (PT/RAM)
$E(x_1)$	4.478	4.4708 (0.091)	4.5019 (0.107)	4.4185 (0.170)	1.44	3.89
$E(x_2)$	4.905	4.9318 (0.101)	4.9439 (0.139)	4.8790 (0.283)	1.91	7.40
$E(x_1^2)$	25.605	25.5717 (0.900)	25.9241 (1.098)	24.9856 (1.713)	1.61	4.09
$E(x_2^2)$	33.920	33.2234 (1.100)	34.4763 (1.373)	33.5966 (2.867)	1.69	6.39
Case (b)	Truth	RAM	EE	PT	MSE ratio (EE/RAM)	MSE ratio (PT/RAM)
$E(x_1)$	4.688	4.673 (0.026)	4.699 (0.072)	4.709 (0.116)	5.89	15.42
$E(x_2)$	5.030	5.029 (0.035)	5.037 (0.086)	5.001 (0.134)	6.07	15.33
$E(x_1^2)$	25.558	25.508 (0.263)	25.693 (0.739)	25.813 (1.122)	7.87	18.47
$E(x_2^2)$	31.378	31.456 (0.334)	31.433 (0.839)	31.105 (1.186)	6.01	12.59

chosen pair of chains under adjoining temperature levels at the end of each iteration. We determine the length of each chain and the burn-in size for Metropolis and PT by taking into account their average evaluation cost, denoted by N_π^{M} and N_π^{PT}, respectively[3]. For example, the length of each chain for Metropolis is $500{,}000 \times N_\pi^{\mathrm{RAM}}/N_\pi^{\mathrm{M}}$ and that for PT is $500{,}000 \times N_\pi^{\mathrm{RAM}}/N_\pi^{\mathrm{PT}}$ so that the (expected) total number of target density evaluations is the same for each algorithm. We need to adjust the burn-in size by the average evaluation cost for a fair comparison because a large burn-in size improves the effectiveness of the one-time adaptation.

We use two numerical measures to evaluate each algorithm. The first is the average number of the unknown modes that are discovered by each chain; we denote this by N_{dis} (≤ 6). The second is the average frequency error rate (Kou, Zhou, and Wong 2006), denoted by $F_{\mathrm{err}} = \sum_{i=1}^{10} \sum_{j=1}^{8} |F_{i,j} - 1/8|/80$, where $F_{i,j}$ is the proportion of iterations in chain i whose nearest mode measured by the Euclidean distance is μ_j.

[3] With a caching strategy, PT evaluates the target once for a Metropolis transition under each of five temperature levels and evaluates it twice for a swap at the end of each iteration.

Table 3 summarizes the results, and shows that using the same jumping rule, RAM is never worse than Metropolis in terms of N_{dis} and F_{err} regardless of dimension, and the improvement on F_{err} can be substantial. It also shows that RAM's F_{err} starts off smaller than that of PT but deteriorates faster than PT's once $d > 5$, and that PT discovers all six modes for every d. This demonstrates the value of fine tuning particularly in higher dimensions for PT, including the number of parallel chains, temperature and proposal scale at each chain, and the number and rate of swaps at each iteration. Therefore, if one can afford the tuning cost, then PT has much to recommend it, especially in high dimensions.

3.3. *Example 3: Sensor Network Localization*

For high-dimensional sampling, a blocked Gibbs sampler (Geman and Geman 1984) is sometimes more convenient and intuitive than direct Metropolis sampling. Here, we consider a realistic example from Ihler et al. (2005): searching for unknown sensor locations within a network using the noisy distance data. This is called sensor network localization (Ihler et al. 2005; Lan, Streets, and Shahbaba 2014). This problem is known to produce

Table 3. The sampling results include the length of each chain before discarding burn-in; the number of burn-in iterations; N_π = the average number of target density evaluations at each iteration; N_d = the average number of downhill proposals for RAM; N_u = the average number of uphill proposals for RAM; N_z = the average number of downhill proposals for the auxiliary variable for RAM; acceptance rate; N_{dis} = the average number of the unknown modes that are discovered by each chain; and $F_{\text{err}} = \sum_{i=1}^{10}\sum_{j=1}^{8}|F_{i,j} - 1/8|/80$, where $F_{i,j}$ is the proportion of iterations in chain i whose nearest mode is μ_j.

d	Kernel	Length of a chain (burn-in size)	N_π (N_d, N_u, N_z)	Acceptance rate	N_{dis}	F_{err}
3	Metropolis	3,272,000 (1,308,800)	1	0.036	6.0	0.021
	PT	467,429 (186,971)	7	0.025	6.0	0.025
	RAM	500,000 (200,000)	6.544 (1.014, 4.210, 1.320)	0.101	6.0	0.019
5	Metropolis	3,768,500 (1,507,400)	1	0.019	6.0	0.047
	PT	538,357 (215,343)	7	0.041	6.0	0.041
	RAM	500,000 (200,000)	7.537 (1.009, 5.222, 1.306)	0.052	6.0	0.038
7	Metropolis	4,220,500 (1,688,200)	1	0.014	5.8	0.209
	PT	602,929 (241,171)	7	0.058	6.0	0.058
	RAM	500,000 (200,000)	8.441 (1.006, 6.136, 1.299)	0.036	6.0	0.075
9	Metropolis	4,734,000 (1,893,600)	1	0.012	5.6	0.312
	PT	676,286 (270,514)	7	0.075	6.0	0.075
	RAM	500,000 (200,000)	9.468 (1.005, 7.171, 1.292)	0.029	5.7	0.182
11	Metropolis	5,350,000 (2,140,000)	1	0.023	5.3	0.512
	PT	764,286 (305,714)	7	0.004	6.0	0.108
	RAM	500,000 (200,000)	10.700 (1.003, 8.416, 1.281)	0.021	5.5	0.267

a high-dimensional, banana-shaped, and multimodal joint posterior distribution.

Modifying Lan, Streets, and Shahbaba (2014)'s simulation setting[4], we suppose there are six stationary sensors scattered on a two dimensional space, and let $x_k^\top = (x_{k1}, x_{k2})$ denote the two-dimensional coordinates of the location of sensor k for $k = 1, 2, \ldots, 6$. We assume that the locations of the last two sensors, x_5 and x_6, are known and the locations of the other sensors, x_1, x_2, x_3, and x_4, are unknown parameters of interest. The Euclidean distance between two sensors, x_i and x_j, denoted by y_{ij} $(= y_{ji})$, is observed with a distance-dependent probability and Gaussian measurement error for $i = 1, 2, \ldots, 5$ and $j = i+1, \ldots, 6$. The probability distributions for the observed data are

$$w_{ij} \mid x_1, \ldots, x_4 \sim \text{Bernoulli}\left(\exp\left(-\frac{\|x_i - x_j\|^2}{2 \times 0.3^2}\right)\right)$$

and

$$y_{ij} \mid (w_{ij} = 1),\ x_1, \ldots, x_4 \sim \text{N}_1\left(\|x_i - x_j\|,\ 0.02^2\right),$$

where w_{ij} $(= w_{ji})$ is an indicator variable that equals one if the distance between x_i and x_j is observed. Simulated distances y_{ij} are displayed in Figure 4, where $w_{ij} = 1$ if y_{ij} is specified and zero otherwise. For each unknown location, we assume a diffuse bivariate Gaussian prior distribution with mean (0, 0) and covariance matrix $10^2 \times I_2$. The eight-dimensional likelihood function is thus

$$L(x_1, x_2, x_3, x_4) \propto \prod_{j>i}\left[\exp\left(-\frac{(y_{ij} - \|x_i - x_j\|)^2}{2 \times 0.02^2}\right) \times \exp\left(-\frac{w_{ij} \times \|x_i - x_j\|^2}{2 \times 0.3^2}\right) \times \left(1 - \exp\left(-\frac{\|x_i - x_j\|^2}{2 \times 0.3^2}\right)\right)^{1-w_{ij}}\right]$$

and the full posterior distribution is

$$\pi(x_1, x_2, x_3, x_4 \mid y, w) \propto L(x_1, x_2, x_3, x_4) \times \exp\left(-\frac{\sum_{k=1}^4 x_k^\top x_k}{2 \times 10^2}\right), \quad (12)$$

where $y = \{y_{ij}, i > j\}$ and $w = \{w_{ij}, i > j\}$. This model may suffer from nonidentifiability when the number of observed

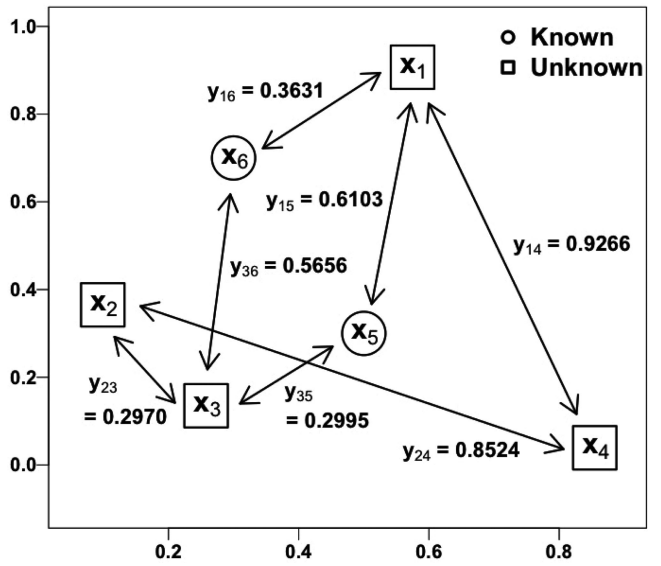

Figure 4. The simulated distances y_{ij} $(= y_{ji})$ among the six stationary sensor locations, $x_1, x_2, \ldots, x_6$, are displayed if observed. The observation indicator w_{ij} $(= w_{ji})$ is one if y_{ij} is specified and is zero otherwise. The locations of the sensors are $x_1 = (0.57, 0.91)$, $x_2 = (0.10, 0.37)$, $x_3 = (0.26, 0.14)$, $x_4 = (0.85, 0.04)$, $x_5 = (0.50, 0.30)$, and $x_6 = (0.30, 0.70)$, where the first four locations, x_1, x_2, x_3, and x_4, are assumed to be unknown.

[4] We remove some locations and adjust observed distances to make a simpler model, yet the resulting posterior distributions have more complicated shapes.

distances is small because unknown locations appear in the likelihood only through distances; if y_{ij} is observed between an unknown x_i and a known x_j, the posterior distribution of x_i may form a circle around x_j without further observations.

We sample (12) using a Gibbs sampler by iteratively sampling the four bivariate conditionals denoted by $\pi_i(x_i \mid x_j, j \neq i, y, \omega)$ for $i = 1, 2, 3, 4$. Since none of these is a standard distribution, we use Metropolis, RAM, or tempered transition (TT) (Neal 1996) kernels that are invariant with respect to each conditional distribution; see Appendix B for details of TT, jumping rules, and initial values. To sample x_k from a RAM kernel that is marginally invariant to π_k, we must keep track of the auxiliary variable during the run, that is, $\{z_k^{(i)}, i = 0, 1, 2, \ldots\}$. At iteration i, we sequentially draw $x_k' \sim q^{\mathrm{D}}(x_k' \mid x_k^{(i-1)})$, $x_k^* \sim q^{\mathrm{U}}(x_k^* \mid x_k')$, and $z_k^* \sim q^{\mathrm{D}}(z_k^* \mid x_k^*)$. We set $(x_k^{(i)}, z_k^{(i)})$ to (x_k^*, z_k^*) with probability $\alpha^{\mathrm{J}}(x_k^*, z_k^* \mid x_k^{(i-1)}, z_k^{(i-1)})$ given in (8), and set $(x_k^{(i)}, z_k^{(i)})$ to $(x_k^{(i-1)}, z_k^{(i-1)})$ otherwise. Because $\{z_k^{(i)}, i = 0, 1, 2, \ldots\}$ are introduced solely to enable sampling x_k from a RAM kernel, only $x_k^{(i)}$ is used to sample the other locations, and $z_k^{(i)}$ is used to draw $x_k^{(i+1)}$ at the next iteration.

For a fair comparison, we set the length of each chain to have the same average number of evaluations of π_i's per iteration. As before, we use $\mathrm{N}_\pi^{\mathrm{X}}$ to denote the average evaluation cost. We first implement RAM within a Gibbs sampler for 220,000 iterations with the first 20,000 as burn-in, resulting in $N_\pi^{\mathrm{RAM}} = 36.13$, that is, about nine density evaluations are required to sample each of the π_i's (with caching). Since $N_\pi^{\mathrm{M}} = 4$ and $N_\pi^{\mathrm{TT}} = 24$ (with caching), we set the length of each Metropolis chain and TT chain respectively to $220{,}000 \times N_\pi^{\mathrm{RAM}}/N_\pi^{\mathrm{M}}$ and $220{,}000 \times N_\pi^{\mathrm{RAM}}/N_\pi^{\mathrm{TT}}$. However, unlike the previous example where there is a one-time adaption and hence it is important to adjust for the burn-in length as well, here we discard the first 20,000 iterations as burn-in for all three algorithms. This burn-in size is sufficient to remove the effect of random initial values of the algorithms.

Table 4. The sampling results summarize the length of each chain (including the 20,000 burn-in iterations); N_π = the average number of evaluating π_1, π_2, π_3, and π_4 at each iteration; details of N_π for each location; and the acceptance rates.

Kernel	Length of a chain	N_π	Details of N_π (N_d, N_u, N_z)	Acceptance rate
Metropolis	1,987,150	$N_\pi^{\mathrm{M}} = 4$	1 for each of $x_1, \ldots, x_4$	0.00057 for x_1 0.00151 for x_2 0.00053 for x_3 0.00115 for x_4
Tempered transitions	331,192	$N_\pi^{\mathrm{TT}} = 24$	6 for each of $x_1, \ldots, x_4$	0.00360 for x_1 0.01034 for x_2 0.00369 for x_3 0.00918 for x_4
RAM	220,000	$N_\pi^{\mathrm{RAM}} = 36.13$	9.40 for x_1 (1, 7.33, 1.07) 8.64 for x_2 (1, 6.56, 1.08) 9.22 for x_3 (1, 7.16, 1.06) 8.87 for x_4 (1, 6.74, 1.13)	0.00349 for x_1 0.00830 for x_2 0.00353 for x_3 0.00730 for x_4

Table 4 summarizes the configurations of the samplers and their acceptance rates. RAM improves the acceptance rate of Metropolis by a factor at least of 5.5 given the same jumping rule without additional tuning. TT improves the acceptance rates even further by a factor of at least 6.3 (relative to Metropolis), but it requires additional tuning of the number of temperature levels, temperature, and jumping scale at each temperature level.

Figure 5 gives scatterplots of the posterior samples of each unknown sensor location (rows) obtained by the three samplers

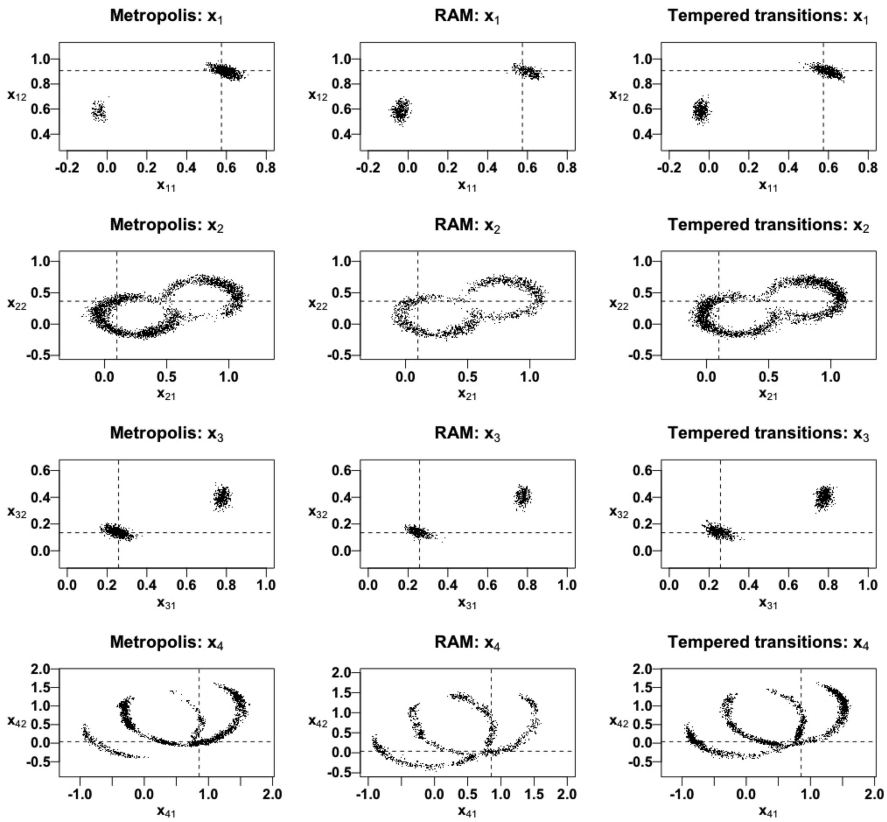

Figure 5. Scatterplots of the posterior sample of each location (rows) obtained by different samplers (column). The coordinates of the unknown sensors are denoted by dashed lines.

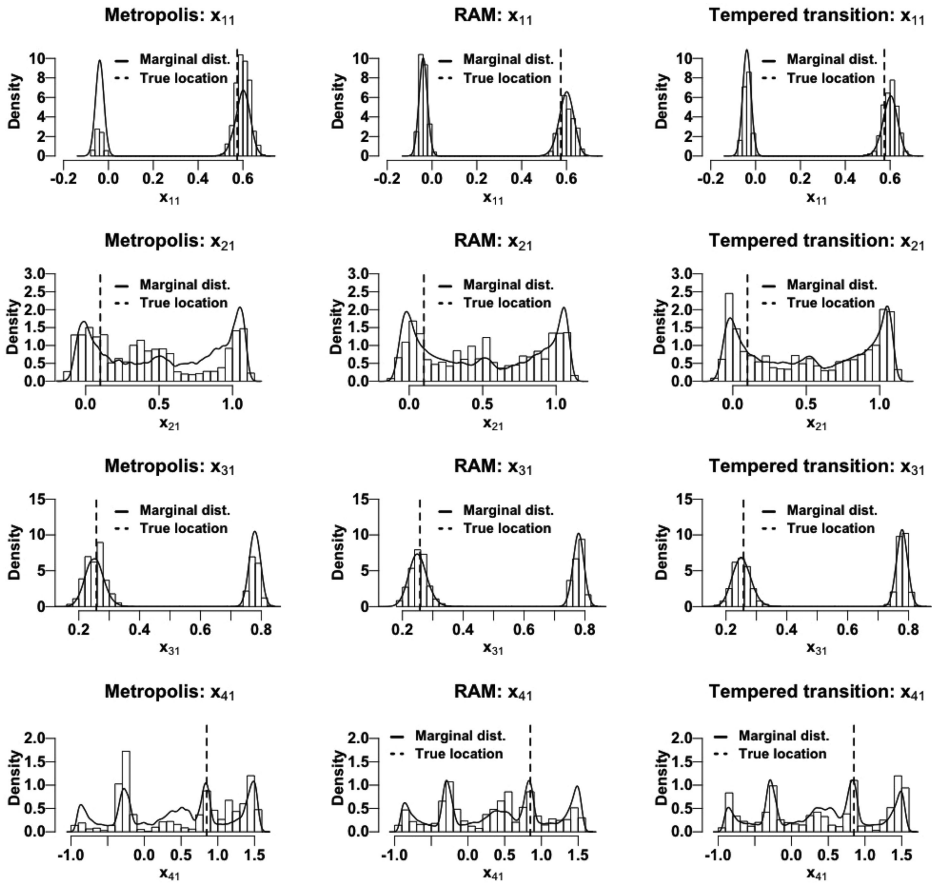

Figure 6. Histograms of the posterior sample of each first coordinate (rows) obtained by different kernels (columns). In each histogram, the marginal posterior density based on 20 million posterior samples obtained with each sampler is superimposed. The vertical dashed lines indicate the true sensor locations.

(columns), where the dashed lines indicate the coordinates of the true location. The RAM sample is more dispersed than that of Metropolis, especially for x_1, x_2, and x_4, with the same jumping rule and is as dispersed as that of TT without subtle tuning that TT requires. The posterior samples of both Metropolis and TT, however, are denser than that of RAM because of their larger sample sizes.

Figure 6 compares the relative sizes of modes for the first coordinate of each unknown location (rows) obtained by each sampler (columns). In each histogram, we superimpose the marginal posterior density based on twenty million posterior draws obtained from each sampler after confirming that the shapes of the posterior densities obtained in this manner are almost identical for the three algorithms. The vertical dashed lines indicate the true sensor locations. RAM represents all four distributions better than Metropolis does, and it does as well as TT, but without the tuning requirement of the latter.

3.4. Example 4: Strong Lens Time Delay Estimation

Our final numerical illustration targets a multimodal distribution, where one mode is extremely distant from the others. This multimodal distribution arises from the applied astrophysical problem that originally motivated the development of RAM; see Tak et al. (2017) for details. Here, we review the problem and discuss a new efficient algorithm.

When there is a massive galaxy between a highly luminous quasar and the Earth, the gravitational field of the galaxy may act as a lens, bending the light emitted by the quasar. This may produce two (or more) slightly offset images of the quasar, an effect known as strong gravitational lensing (Schneider, Wambsganss, and Kochanek 2006). There may be a time delay between the images in that their light follows different paths with different travel times. Thus, temporal features in time series of the brightness of each image appear shifted in time. The time delay is, for example, used to calculate the current expansion rate of the Universe, that is, the Hubble constant (Refsdal 1964).

Figure 7 displays two irregularlyobserved time series of the brightness of the doublylensed quasar Q0957 + 561 (Hainline et al. 2012); the two time series are labeled *A* and *B*. Brightness is reported on a magnitude scale where smaller values correspond to brighter images. Let $x \equiv \{x_1, \ldots, x_n\}$ and $y \equiv \{y_1, \ldots, y_n\}$ denote the n magnitudes irregularly observed at

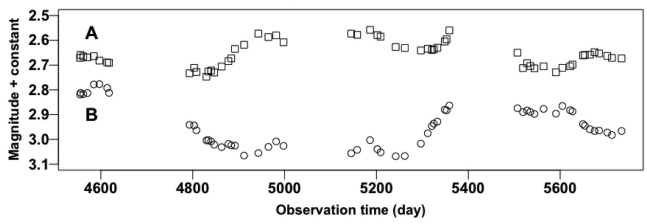

Figure 7. Two observed time series of doublylensed quasar Q0957+561 (Hainline et al. 2012). Time series *A* is denoted by squares and time series *B* is denoted by circles. Magnitude is an astronomical measure of brightness. Both time series are plotted with an offset (constant) in magnitude, but this does not affect the time delay estimation.

time $t \equiv \{t_1, \ldots, t_n\}$ in time series A and B, respectively. Let $\delta \equiv \{\delta_1, \ldots, \delta_n\}$ and $\eta \equiv \{\eta_1, \ldots, \eta_n\}$ represent the n known measurement standard deviations for x and y, respectively. There are 57 observations in each time series, that is, $n = 57$.

We assume that for each observed time series there is an unobserved underlying brightness curve. Let $X(t) \equiv \{X(t_1), \ldots, X(t_n)\}$ denote the latent magnitudes for time series A and $Y(t) \equiv \{Y(t_1), \ldots, Y(t_n)\}$ denote those for time series B. We further assume that one of the latent brightness curves is a shifted version of the other, that is,

$$Y(t_j) = X(t_j - \Delta) + \beta_0, \tag{13}$$

where Δ is the unknown time delay and β_0 is an unknown magnitude offset.

The observed magnitudes given the latent magnitudes are assumed to be independent Gaussian variables:

$$x_j \mid X(t_j) \sim N_1\big(X(t_j),\ \delta_j^2\big) \ \text{ and } \ y_j \mid Y(t_j) \sim N_1\big(Y(t_j),\ \eta_j^2\big). \tag{14}$$

Using (13), we can express the model for y in (14) as

$$y_j \mid X(t_j - \Delta),\ \Delta,\ \beta_0 \sim \mathrm{N}_1\big(X(t_j - \Delta) + \beta_0,\ \eta_j^2\big). \tag{15}$$

We assume $X(\cdot)$ follows an Ornstein–Uhlenbeck process (Kelly, Bechtold, and Siemiginowska 2009). Solving the resulting stochastic differential equation yields the sampling distribution of $X(t^\Delta)$, where $t^\Delta \equiv (t_1^\Delta, \ldots, t_{2n}^\Delta)^\top$ contains the sorted $2n$ times among the n observation times, t, and the n time-delay-shifted observation times, $t - \Delta$. Specifically,

$$\begin{aligned} X\left(t_1^\Delta\right) \mid \Delta, \theta &\sim N_1\left(\mu, \frac{\tau\phi^2}{2}\right), \text{ and for } j = 2, 3, \ldots, 2n, \\ X\left(t_j^\Delta\right) \mid X(t_{j-1}^\Delta), \Delta, \theta &\sim N_1\left(\mu + a_j\big(X(t_{j-1}^\Delta) - \mu\big), \frac{\tau\phi^2}{2}\left(1 - a_j^2\right)\right), \end{aligned} \tag{16}$$

where $\theta \equiv (\mu, \phi^2, \tau)^\top$ and $a_j = \exp(-(t_j^\Delta - t_{j-1}^\Delta)/\tau)$.

Following Tak et al. (2017), we set independent priors for the model parameters:

$$\begin{aligned} \Delta &\sim \text{Uniform}[-1178.939,\ 1178.939], \\ \beta_0 &\sim \text{Uniform}[-60, 60], \\ \mu &\sim \text{Uniform}[-30, 30],\ \phi^2 \sim \text{inverse-Gamma}(1, 2/10^7), \\ \tau &\sim \text{inverse-Gamma}(1, 1). \end{aligned} \tag{17}$$

The resulting joint posterior density function is

$$\begin{aligned} &\pi(X(t^\Delta), \Delta, \beta_0, \theta \mid x, y) \\ &\quad \propto \left[\prod_{j=1}^{n} f_1\left(x_j \mid X(t_j)\right) \times f_2\left(y_j \mid X(t_j - \Delta), \Delta, \boldsymbol{\beta}\right)\right] \\ &\quad \times g\left(X\left(t_1^\Delta\right) \mid \Delta, \boldsymbol{\theta}\right) \times \left[\prod_{j=2}^{2n} g\left(X\left(t_j^\Delta\right) \mid X\left(t_{j-1}^\Delta\right), \Delta, \boldsymbol{\theta}\right)\right] \\ &\quad \times h(\Delta, \beta_0, \theta), \end{aligned} \tag{18}$$

where the density functions, f_1, f_2, g, and h are defined by (14)–(17), respectively.

To sample from (18), we adopt an MH within Gibbs sampler (Tierney 1994) composed of the three steps shown in Table 5; see Appendices A–C of Tak et al. (2017) for details. We suppress conditioning on x and y here and elsewhere. Because we cannot directly sample $\pi_{11}(\Delta \mid \beta_0, \theta)$ in *Step* 1 and the marginal posterior distribution of Δ is often multimodal, we draw Δ using one of four algorithms: (i) Metropolis, (ii) Metropolis with a mixture jumping rule, (iii) RAM, or (iv) TT. The mixture jumping rule generates a proposal from the Gaussian jumping rule used by Metropolis with probability 0.5 and from the prior distribution of Δ otherwise. To sample Δ using the RAM kernel, we additionally keep track of the auxiliary variable during the run, that is, $\{z^{(i)}, i = 0, 1, 2, \ldots\}$. At iteration i, we sequentially draw $\Delta' \sim q^{\mathrm{D}}(\Delta' | \Delta^{(i-1)})$, $\Delta^* \sim q^{\mathrm{U}}(\Delta^* | \Delta')$, and $z^* \sim q^{\mathrm{D}}(z^* | \Delta^*)$. We set $(\Delta^{(i)}, z^{(i)})$ to (Δ^*, z^*) with acceptance probability $\alpha^{\mathrm{J}}(\Delta^*, z^* \mid \Delta^{(i-1)}, z^{(i-1)})$ given in (8), and set $(\Delta^{(i)}, z^{(i)})$ to $(\Delta^{(i-1)}, z^{(i-1)})$ otherwise. Because $\{z^{(i)},\ i = 0, 1, 2, \ldots\}$ are introduced solely to enable sampling Δ from the RAM kernel, only $\Delta^{(i)}$ is used to sample $X(t^\Delta)$, β_0, and θ in the other steps in Table 5, and $z^{(i)}$ is used to draw $\Delta^{(i+1)}$ at the next iteration.

Specifically, we fit the time delay model using the MH within Gibbs sampler equipped with TT for Δ first, initiating a single long chain of length 5,050,000 at the center of the entire range of Δ, that is, $\Delta^{(0)} = 0$. We set the initial values of the other parameters as follows: $\beta_0^{(0)} = \sum_{j=1}^n \{y_j - x_j\}/n = -0.113$, $\mu^{(0)} = \sum_{j=1}^n x_j/n = 2.658$, $\phi^{(0)} = 0.01$, $\tau^{(0)} = 200$, and $X^{(0)}(t^{\Delta^{(0)}})$ is a vector of x and $y - \beta_0^{(0)}$ that are sorted in time, t for x and $t - \Delta$ for $y - \beta_0^{(0)}$. Multiple initial values spread across the entire range result in nearly identical posterior distributions. We discard the first 50,000 draws as burn-in. For the tuning parameters of TT, we set five temperature levels, $T_j = 4^j$ for $j = 1, \ldots, 5$, and corresponding jumping scales for Metropolis updates, $\sigma_j = 500 \times 1.2^{j-1}$, so that σ_5 (= 1037) is about a half of the length of the range of Δ. Using the same initial values ($z^{(0)} = \Delta^{(0)}$ for RAM), we obtain an additional chain using each of the MH within Gibbs sampler equipped with Metropolis, RAM, and Metropolis with a mixture jumping rule. In all these cases, we set q to be Gaussian with $\sigma = 700$, that is, about one-third length of the entire range and similar to the jumping scale of TT at the middle temperature level ($\sigma_3 = 720$). This value of σ should be advantageous to Metropolis because it roughly equals the distance between the modes. Since Metropolis, RAM, and Metropolis with a mixture jumping rule take less CPU time than TT, we run longer chains of the three algorithms to match the CPU time, discarding the first 50,000 iterations of each as burn-in; see Appendix C for details of the average number of π_{11} evaluations.

Table 6 summarizes the results from running each algorithm for nearly the same CPU time (28,352 s). Overall, given the same jumping rule and without additional tuning, RAM improves

Table 5. A Metropolis-Hastings within Gibbs sampler for the time delay model. We draw Δ from a kernel that is invariant to π_{11} and draw $X(t^\Delta)$ from π_{12} if Δ is newly updated.

Set initial values $\Delta^{(0)}$, $X^{(0)}(t^{\Delta^{(0)}})$, $\beta_0^{(0)}$, and $\theta^{(0)}$. For $i = 1, 2, \ldots,$

Step 1: Draw $(X^{(i)}(t^{\Delta^{(i)}}), \Delta^{(i)}) \sim \pi_1(X(t^\Delta), \Delta \mid \beta_0^{(i-1)}, \theta^{(i-1)})$
$= \pi_{11}(\Delta \mid \beta_0^{(i-1)}, \theta^{(i-1)})\pi_{12}(X(t^\Delta) \mid \Delta, \beta_0^{(i-1)}, \theta^{(i-1)})$.

Step 2: Draw $\beta_0^{(i)} \sim \pi_2(\beta_0 \mid \theta^{(i-1)}, X^{(i)}(t^{\Delta^{(i)}}), \Delta^{(i)})$.

Step 3: Draw $\theta^{(i)} \sim \pi_3(\theta \mid X^{(i)}(t^{\Delta^{(i)}}), \Delta^{(i)}, \beta_0^{(i)})$.

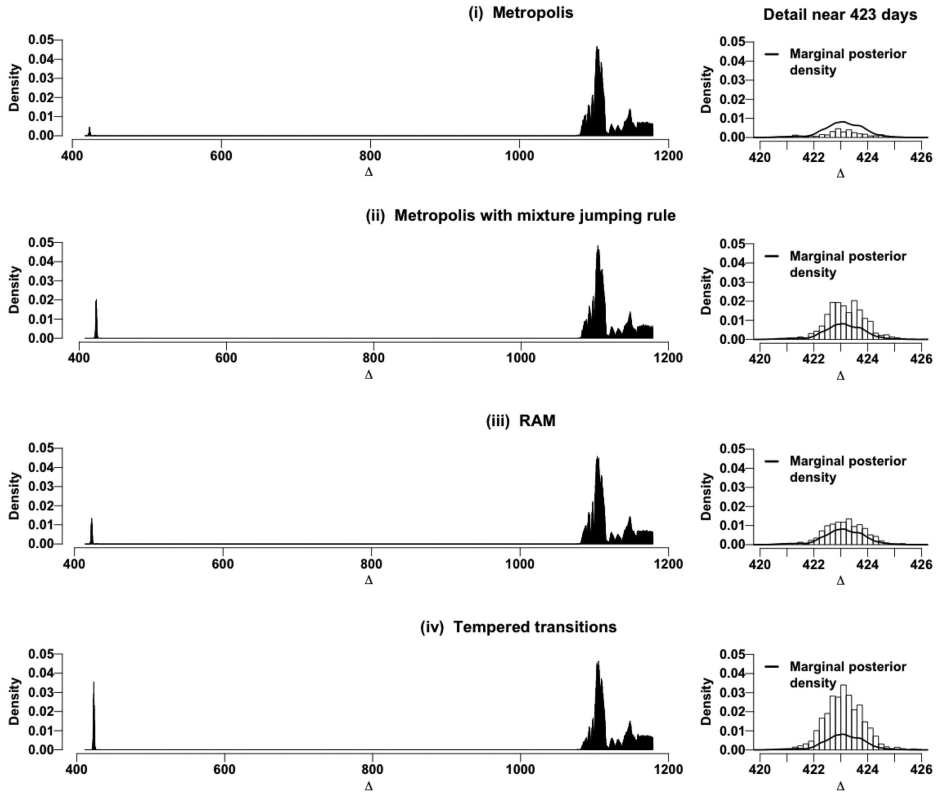

Figure 8. Results of running the algorithms for nearly the same CPU time. The rows represent the four samplers. The first column displays the histograms based on the posterior sample of Δ and the second column focuses on the mode near 423 days. In the second column, we superimpose the posterior density of Δ obtained by an oracle sampler (assuming the mode locations are known) to check the reliability of the relative sizes of the modes.

Table 6. The length of a chain including burn-in; acceptance rate for Δ; and N_{jumps} = the total number of jumps between the two distant modes during the post burn-in run.

	Length of a chain	Acceptance rate	N_{jumps}
(i) Metropolis	22,266,816	0.0150	144
(ii) Metropolis with mixture jumping rule	19,710,188	0.0129	190
(iii) RAM	7,111,612	0.0508	326
(iv) Tempered transitions	5,050,000	0.3022	311

upon both versions of Metropolis; the total number of jumps between the two distant modes in the post burn-in sample, denoted by N_{jumps}, is at least 1.7 times higher for RAM, and RAM's acceptance rate is at least 3.4 times higher. With additional tuning of the number of rungs, temperature, and jumping scale, TT performs no better than RAM in terms of N_{jumps} but its acceptance rate is about 5.9 times higher than Metropolis.

The first column of Figure 8 displays histograms of the posterior sample of Δ obtained using the four different kernels. The size of the mode near 423 days, which is of great scientific interest, differs substantially among the samplers. In the second column of Figure 8, we magnify this mode, superimposing a curve that represents the marginal posterior density of Δ based on twenty million posterior samples obtained with an oracle sampler[5] constructed with knowledge of both mode locations. The size and shape of the mode near 423 days obtained with RAM match the oracle sampler better than the other algorithms, which is an algorithmic confirmation of the reliability of RAM.

4. Concluding Remarks

We propose RAM both as an alternative to deal with multimodality, and as a newer strategy of forming acceptance probabilities. It can also be viewed as using negative temperature in annealing type of strategies, as Professor Art Owen recognized in his comments on an early version of our article.

More work is needed to extend RAM's applicability. In particular, we plan to compare the theoretical convergence rate of our algorithm to others, but this is difficult partially owing to the intractable down-up jumping density, q^{DU}. Also, a better set of strategies for tuning RAM in various multimodal cases needs to be investigated. Different ways to encourage a down-up movement in density may exist, for example, mixing anti-Langevin and Langevin algorithms or tempering with negative and positive temperature levels, both of which were suggested in a personal blog of Professor Christian P. Robert[6]. Another avenue for further improvement is to apply the ideas of the mode-jumping proposal and the delayed rejection method to RAM, e.g., allowing an asymmetric density function q so that the downhill move encourages longer jumps than the uphill move does. Using this down-up idea to construct a global optimizer is another possible extension as the tempering idea is used for a statistical annealing. We invite interested readers to explore these possibilities.

[5] We use an MH within Gibbs sampler equipped with an independent Metropolis kernel (Tierney 1994) that is invariant to π_{11}. The jumping rule for this kernel is Uniform[400, 450] with probability 0.1 and from Uniform[1050, 1178.939] otherwise. We emphasize that this algorithm would not be feasible without prior knowledge of the size and location of the two posterior modes.

[6] *https://xianblog.wordpress.com/2016/01/28/love-hate-metropolis-algorithm/*

Supplementary Materials

Appendices: Appendices A, B, and C as cited in the article (Appendices.pdf).

R code and data: All the R code and data used in this article (RAM.zip).

Acknowledgments

This project was conducted under the auspices of the CHASC International Astrostatistics Center. CHASC is supported by the NSF grants DMS 1208791, 1209232, 1513484, 1513492, and 1513546. Xiao-Li Meng acknowledges partial financial support from the NSF grants given to CHASC. Hyungsuk Tak acknowledges partial support from the NSF grants DMS 1127914 and 1638521 given to SAMSI. David A. van Dyk acknowledges support from a Wolfson Research Merit Award (WM110023) provided by the British Royal Society and from a Marie-Curie Career Integration Grant (FP7-PEOPLE-2012-CIG-321865) provided by the European Commission. The authors thank the associate editor and two referees for their insightful comments and suggestions that significantly improved the presentation. We also thank Christian P. Robert, Pierre E. Jacob, Art B. Owen, and Natesh S. Pillai for very helpful discussions and Steven R. Finch for his proofreading.

Funding

National Science Foundation [DMS 1127914, DMS 1208791, DMS 1209232, DMS 1513484, DMS 1513492, DMS 1513546, DMS 1638521]; Wolfson Research Merit Award [WM110023]; Marie-Curie Career Integration [FP7-PEOPLE-2012-CIG-321865]

ORCID

Hyungsuk Tak http://orcid.org/0000-0003-0334-8742

References

Andrieu, C., and Roberts, G. O. (2009), "The Pseudo-Marginal Approach for Efficient Monte Carlo Computations," *The Annals of Statistics*, 37, 697–725. [481]

Beaumont, M. A. (2003), "Estimation of Population Growth or Decline in Genetically Monitored Populations," *Genetics*, 164, 1139–1160. [481]

Chib, S., and Greenberg, E. (1995), "Understanding the Metropolis-Hastings Algorithm," *The American Statistician*, 49, 327–335. [480]

Geman, S., and Geman, D. (1984), "Stochastic Relaxation, Gibbs distributions, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 721–741. [484]

Geyer, C. J. (1991), "Markov Chain Monte Carlo Maximum Likelihood," *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, Fairfax Station, VA: Interface Foundation of North America, pp. 156–163. [479]

Geyer, C. J., and Thompson, E. A. (1995). "Annealing Markov chain Monte Carlo with Applications to Ancestral Inference," *Journal of the American Statistical Association*, 90, 909–920. [479]

Hainline, L. J., Morgan, C. W., Beach, J. N., Kochanek, C., Harris, H. C., Tilleman, T., Fadely, R., Falco, E. E., and Le, T. X. (2012), "A New Microlensing Event in the Doubly Imaged Quasar Q0957+561," *The Astrophysical Journal*, 744, 104–113. [487]

Hastings, W. K. (1970), "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika*, 57, 97–109. [479]

Ihler, A. T., Fisher, J. W., Moses, R. L., and Willsky, A. S. (2005), "Nonparametric Belief Propagation for Self-Localization of Sensor Networks," *IEEE Journal on Selected Areas in Communications*, 23, 809–819. [484]

Kelly, B. C., Bechtold, J., and Siemiginowska, A. (2009), "Are the Variations in Quasar Optical Flux Driven by Thermal Fluctuations?" *The Astrophysical Journal*, 698, 895–910. [488]

Kou, S. C., Zhou, Q., and Wong, W. H. (2006), "Discussion Paper: Equi-Energy Sampler with Applications in Statistical Inference and Statistical Mechanics," *The Annals of Statistics*, 34, 1581–1619. [479,482,483,484]

Lan, S., Streets, J., and Shahbaba, B. (2014), "Wormhole Hamiltonian Monte Carlo," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2014, 1953–1959. [484,485]

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953), "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, 21, 1087–1092. [479]

Møller, J., Pettitt, A. N., Reeves, R., and Berthelsen, K. K. (2006), "An Efficient Markov Chain Monte Carlo Method for Distributions with Intractable Normalising Constants," *Biometrika*, 93, 451–458. [479,481,482]

Neal, R. M. (1996), "Sampling From Multimodal Distributions Using Tempered Transitions," *Statistics and Computing*, 6, 353–366. [479,486]

R Core Team (2016), *R: A Language and Environment for Statistical Computing*, Vienna, Austria: R Foundation for Statistical Computing. [481]

Refsdal, S. (1964), "The Gravitational Lens Effect," *Monthly Notices of the Royal Astronomical Society*, 128, 295–306. [487]

Schneider, P., Wambsganss, J., and Kochanek, C. (2006), *Gravitational Lensing: Strong, Weak and Micro*, New York: Springer-Verlag. [487]

Tak, H., Mandel, K., van Dyk, D. A., Kashyap, V. L., Meng, X.-L., and Siemiginowska, A. (2017), "Bayesian Estimates of Astronomical Time Delays between Gravitationally Lensed Stochastic Light Curves," *The Annals of Applied Statistics*, 11, 1309–1348. [487,488]

Tierney, L. (1994), "Markov Chains for Exploring Posterior Distributions," *The Annals of Statistics*, 22, 1701–1728. [488]

Tierney, L., and Mira, A. (1999), "Some Adaptive Monte Carlo Methods for Bayesian Inference," *Statistics in Medicine*, 18, 2507–2515. [480]

Tjelmeland, H., and Hegstad, B. K. (2001), "Mode Jumping Proposals in MCMC," *Scandinavian Journal of Statistics*, 28, 205–223. [480]

Trias, M., Vecchio, A., and Veitch, J. (2009), "Delayed Rejection Schemes for Efficient Markov-Chain Monte-Carlo Sampling of Multimodal Distributions," arXiv:0904.2207. [480]