

An adaptive Metropolis algorithm with rejection-based Gaussian proposal-scaling for fast convergence in multimodal parameter spaces *

Graham West [†] and John Wallin [‡]

Abstract. Performing MCMC parameter estimation on complex, nonlinear mathematical models can quickly lead to endless searching through highly multimodal parameter spaces. If one does not know the optimal proposal distribution, the burn-in period can take an excessively long time, due to the Markov chain not properly mixing and becoming trapped near a suboptimal mode. Also, if the model of interest is very computationally expensive, one would prefer to evaluate it as few times as possible. With these challenges in mind, we present in this paper an adaptive Metropolis method RSAP (rejection-scaled adaptive proposal) which can grow and shrink the scale of the Gaussian proposal width (independently for each parameter) based on the number of consecutive rejections the chain has encountered. The method is flexible and robust enough to handle parameter spaces that are both high-dimensional and multimodal (and correlated???). It also solves the issue of the unknown optimal proposal width by adjusting it over time. We will provide a proof of its ergodicity and conclude with comprehensive computational results implementing this method on a number of benchmark optimization functions, comparing its convergence to that of the standard Metropolis method.

Key words. Markov chain Monte Carlo, adaptive methods, parameter estimation

AMS subject classifications. 62M05 (markovian estimation), 62M09 (non-markovian estimation)

1. Introduction. Markov chain Monte Carlo (MCMC) methods provide one of the most widely-used and easily accessible solution techniques to the problem of model parameter estimation (PE). As the name suggests, MCMC algorithms use Markov chains to sample the state space of the model being studied, thus obtaining distributions for all of its relevant parameters. With an ever-growing number of MCMC method types—from interacting forests of chains to delayed rejection techniques which approximate a full model evaluation—a suitable method can be found for virtually every problem.

This paper is outlined in the following way. First, we open with a review of the original Metropolis method, as well as some general notes on how kernel mixing and adaptive methods are able to increase its performance. Next, we will discuss our own adaptive kernel mixing method RSAP and the motivating factors for using it, as well as provide a proof of its ergodicity. Lastly, we will present and interpret results of experiments performed on our method using common optimization benchmark functions (including Gaussian, Rosenbrock, and Ackley functions) in several different dimensionalities, going into detail in comparing our method’s performance with that of standard Metropolis over a range of proposal widths.

*Submitted to the editors DATE.

Funding: This work was funded by the Fog Research Institute under contract no. FRI-454.

[†]Department of Computational Science, Middle Tennessee State University, Murfreesboro, TN (gtw2i@mtmail.utsu.edu).

[‡]Department of Computational Science, Middle Tennessee State University, Murfreesboro, TN (jwallin@mts.edu).

1.1. Metropolis. One of the most well-known and widely-used MCMC techniques is the original Metropolis method. First developed by Metropolis et al. [3] and later generalized by Hastings [2], this simple method has been the foundation for countless new and improved versions that have continued to build on the basic principles found within their predecessor.

Given data that we wish for the model to reproduce, the Metropolis method attempts to sample a distribution over the model parameters called the *posterior*. This distribution will have a peak at the values of the model parameters which best approximate the data. However, we do not know the posterior since we cannot directly invert the parameter-data relationship (and since to do so would be to know the solution to the problem). Therefore, we must express it in terms of distributions which we do know. From Bayes theorem, we have

$$(1.1) \quad \pi(\theta|y) \propto \ell(y|\theta)p(\theta)$$

where π is the *posterior* distribution, ℓ is the *likelihood* distribution, and p is the *prior* distribution. As opposed to the posterior—which tells us the probability of the model parameters given the data—the likelihood $\ell(y|\theta)$ tells us the probability of the data given the parameters. Therefore, we can easily calculate its value via a single model evaluation with the given parameters θ (referred to as the state). In addition, we may also have prior information about what parameter values are most likely to be realized in the actual system (for example, if the parameters can only take on values in a finite range or tend to be concentrated about a central mean). This information is incorporated via the prior distribution p . Together, the likelihood and prior help us to reconstruct the posterior distribution (up to a scale factor which does not impact the method) so that the inverse problem may be solved.

Now, concerning the specifics of how the Metropolis method successfully samples the posterior, suppose we're given some data y and an initial state θ_0 , which is a reasonable guess for the proper values of the parameters. The method generates new candidate states and probabilistically accepts them based on their relative improvement from the current state, using the Bayesian approximation of the posterior as a metric. Candidate state generation is performed via sampling of the *proposal* distribution $q(\theta_2|\theta_1)$, defined as the probability that a new state θ_2 will be selected as a candidate for acceptance given the current state θ_1 . Whether acceptance or rejection is performed is determined by the *acceptance probability* $\alpha(\theta_2|\theta_1)$, defined as the probability that the candidate state θ_2 will be accepted given the current step θ_1 . Over time, the distribution created from the samples will conform to the posterior, at which point the method is said to have *converged*. Below is a pseudocode implementation of Metropolis (Algorithm 1.1) which runs for N steps.

Metropolis has several useful properties which contribute its great effectiveness. First—interpreting the method as an annealing-esque optimization method—it can both naively hill-climb and tactically accept poorer states to escape suboptimal local modes. When testing a new candidate state for acceptance, if it has an equal or higher posterior value than the current state, then $\alpha = 1$ and acceptance is guaranteed (hill-climbing). On the other hand, if the posterior is lower, then $1 > \alpha \geq 0$ and the poorer state still has a probability of being accepted. Though poorer states can be accepted, α decreases as the candidate's posterior becomes lower, so very poor candidates will usually be rejected. Counter-intuitively, this ability to accept poorer states is required to guarantee the eventual convergence of Metropolis

Algorithm 1.1 Metropolis-Hastings

```

1: Initialize  $\theta_0$ 
2: for  $n = 1$  to  $N$  do
3:   Generate  $\theta'_n \sim q(\theta'_n | \theta_{n-1})$ 
4:   Compute  $\alpha(\theta'_n | \theta_{n-1}) = \min \left( 1, \frac{\pi(\theta'_n | y)}{\pi(\theta_{n-1} | y)} \right)$ 
       $= \min \left( 1, \frac{\ell(y | \theta'_n) p(\theta'_n)}{\ell(y | \theta_{n-1}) p(\theta_{n-1})} \right)$ 
5:   Set  $\theta_n = \theta'_n$  with probability  $\alpha$ , else  $\theta_n = \theta_{n-1}$ 
6: end for
    
```

78 (as evidenced by the appearance of the acceptance probability in the proof in the following
 79 paragraph).

80 This leads us to the second useful property of Metropolis. As long as the proposal distri-
 81 bution endows the Markov chain with a few key properties (irreducibility, aperiodicity, and
 82 non-transience)—which hold for nearly all reasonable proposals—the distribution will always
 83 converge to the posterior, the stationary distribution of the chain formed by Metropolis. As-
 84 suming the three conditions mentioned above hold for the chain, we can easily show that the
 85 posterior distribution is the stationary distribution of the Metropolis method (proof taken
 86 from Gelman [4]). Say the algorithm is at the n -th step and that $\theta_n = a$, where a was drawn
 87 from the posterior. Now, suppose we propose the candidate $\theta_{n+1} = b$ such that $\pi(b) \geq \pi(a)$.
 88 Then,

$$89 \quad (1.2) \quad P(\theta_n = a, \theta_{n+1} = b) = \min \left(1, \frac{\pi(b)}{\pi(a)} \right) q(b|a) \pi(a) = q(b|a) \pi(a)$$

90 On the other hand, suppose $\theta_n = b$ and we attempt a transition to the candidate $\theta_{n+1} = a$.
 91 Then,

$$92 \quad (1.3) \quad \begin{aligned} P(\theta_n = b, \theta_{n+1} = a) &= \min \left(1, \frac{\pi(a)}{\pi(b)} \right) q(a|b) \pi(b) \\ &= \frac{\pi(a)}{\pi(b)} q(a|b) \pi(b) = q(a|b) \pi(a) = q(b|a) \pi(a) \end{aligned}$$

93 Therefore, $P(\theta_n = a, \theta_{n+1} = b) = P(\theta_n = b, \theta_{n+1} = a)$, which is the condition for stationarity.

94 Note that since the stationary distribution—being simply the posterior—is independent
 95 of the proposal used, and thus holds for all valid proposals. However, though the method is
 96 guaranteed to always converge to π , it may take a large number of steps for convergence to
 97 occur. Much of the algorithm’s utility rests on the choice of a good proposal q . If one chooses
 98 too thin of a distribution, the method may have difficulty escaping suboptimal modes within
 99 an acceptable amount of time. On the other hand, too wide a distribution and the method
 100 might jump past true optimal modes or begin to oscillate about them, taking a large number
 101 of steps to sink into them.

With this in mind, the key question is, "What is the optimal proposal to use for problem X ?" As should be expected, to know the answer to this question is essentially to know the solution to problem X . Therefore, certain assumptions and compromises must be made when designing proposals. Fortunately, as shall be seen in the following sections, this problem of the unknown proposal can be largely alleviated through clever means such as kernel mixing and adaptive proposals.

1.2. Kernel mixing and composition. Kernel mixing and composition are techniques where the transition probabilities (or kernel) used at each step in the chain may be different. Specifically, *composition* refers to a method where the kernel is chosen at each step through some deterministic means. Consider, for example, the simple case we have two kernels P_1, P_2 and simply alternate between them, producing the n -step kernel $P_1 P_2 P_1 P_2 \cdots P_k$, where $k = 1$ if n is odd and $k = 2$ if n is even (note that we use the convention commonly found in Markov chain literature of representing states as row vectors and right multiplying them by the kernel).

On the other hand, techniques where the kernels used at each step are determined by some random process are termed *mixing* methods. Consider another example where we have two kernels P_1, P_2 , but instead of strictly alternating, we have a probability 0.5 of choosing either at any given step. Then, the kernel used at each step is simply $0.5P_1 + 0.5P_2$ —the linear combination of the kernels and their respective probabilities—and the n -step kernel is $(0.5P_1 + 0.5P_2)^n$. Upon an actual run of this method, the true n -step kernel used is $P_{i_1} P_{i_2} \cdots P_{i_n}$ where the i_j are determined by performing some random process on the given distribution. However, when run for a large number of steps, the second kernel will converge to the first. As shall be seen in section 2, our method uses kernel mixing by randomly selecting between numerous Gaussian proposals with varying widths.

1.3. Adaptive MCMC. Another technique for improving the performance of MCMC is to use so-called *adaptive* methods which allow the kernel to intelligently adapt to the state space as more and more samples are obtained. The chains produced by these methods are referred to as *inhomogeneous* chains since the kernel is not constant over time. For many models with nonlinear interactions between the parameters, neighboring regions in state space can have completely different posterior profiles, causing what would be an optimal proposal in one region to perform poorly in another. There may be numerous local modes clustered together or large plateaus. These effects combined with a lack of prior knowledge of which proposal should be chosen can inevitably lead to long run times. Thus, in these cases, one can justify the use of methods which can tune themselves based on the changing geometry of the state space.

Though adaptive methods can dramatically increase computational performance and efficiency, they often lose some of the desired analytical properties that their more simple counterparts have. For example, if the evolution of the kernel depends on the samples from multiple previous steps, the chain is no longer *Markovian*. Also, in general, one cannot prove ergodicity without the assumption of several special conditions which may be difficult to verify in practice. Fortunately, even if one cannot verify convergence properties for a specific adaptive MCMC method, it may still be used to dramatically shorten the burn-in time as the chain searches for the target distribution. Despite the difficulties discussed, we were actually able to prove our own method's ergodicity through simple (yet notationally dense) means

thanks to Theorem 5 from a paper by Roberts and Rosenthal [8]. See this resource for a very thorough analysis on conditions for the convergence of adaptive MCMC methods.

Two excellent examples of adaptive MCMC methods come from Haario, Saksman, and Tamminen in the form of the *Adaptive Proposal* (AP) [5] and *Adaptive Metropolis* (AM) [6] methods. While both methods are based on the same principle of adaptation, AM retains ergodicity while AP does not. The core principle shared by the two is the use of a continuously adapting Gaussian distribution as the proposal for the standard Metropolis method. Specifically, the proposal used is

$$(1.4) \quad q_n(\theta_2|\theta_1) = N(\theta_1, \left(\frac{2.38^2}{d}\right)\Sigma_n)$$

where Σ_n is the covariance matrix computed from samples generated over past steps. This allows the Gaussian to continually re-scale and re-orient itself as it encounters different regions of state space. Now, the only difference between AP and AM is that AP uses a fixed number of samples to calculate Σ_n while AM uses the entire history of the chain. Thus, AP’s adaptation does not vanish since its covariance matrix does not converge, while AM’s does (due to the central limit theorem).

2. Rejection-scaled adaptive proposal (RSAP). We now move on to present our own contribution to the field of adaptive MCMC: the rejection-scaled adaptive proposal method (RSAP). Based on the original Metropolis method, this method uses very simple means to scale the entries of the Gaussian proposal’s diagonal covariance matrix (i.e., the proposal widths in the direction of each model parameter) to more locally optimal values. We will begin with a discussion of our motivation for developing the method, the challenges it addresses, and some advantages it has over standard Metropolis and some other adaptive methods. We will then move on to a detailed description of the method on a step-by-step level and end with our ergodicity proof.

2.1. Motivation and advantages. There are two main advantages RSAP has over standard Metropolis or other adaptive methods. First, as we have discussed, it isn’t possible in general to know a priori what the optimal proposal width should be for a given problem. Even if one chooses the optimal proposal width, this by no means guarantees that certain regions of state space might not be more efficiently traversed with a different proposal width. These issues are resolved by the ability of RSAP to both grow and shrink the proposal width at a set amplitude and rate, allowing the chain to escape local modes more easily and explore flatter regions more quickly.

Another advantage this method has over other more complicated adaptive methods is its trivial implementation and the minimal overhead it requires. Both the extra storage and computing time required for our method compared to standard Metropolis are negligible, since only several additional tuning parameters, counters, and scale factors must be stored and the adaptive scaling functions are computationally trivial to evaluate. Because of this, there is essentially no reason not to use RSAP as opposed to standard Metropolis.

2.2. The method. The key to the RSAP method is that it mixes a fixed-width Gaussian with two adaptable wide and thin Gaussians (one of each) into a single proposal. The thin

option is included so that the chain can squeeze into tight optimal modes and the wide option so that the chain can both escape suboptimal modes and traverse state space quickly. The fixed option is included so that we always have a reasonable proposal to use in case the others adapt too much to be useful. With this reasoning in mind, the proposal for a single model parameter has the form of the linear combination

$$(2.1) \quad q^n(\theta_{n+1}|\theta_n) = p_t^n N(\theta_n, \sigma_t^2) + p_f^n N(\theta_n, \sigma_f^2) + p_w^n N(\theta_n, \sigma_w^2)$$

where $\sigma_t < \sigma_f < \sigma_w$. Every parameter uses separately adapting wide and thin proposal widths—allowing for independent adaptation of each—as well as its own fixed width (since the different parameters are likely different units/scales).

Concerning how we choose between the three options, we have the time-dependent probabilities p_t^n, p_f^n, p_w^n that either the thin, fixed, or wide proposal (respectively) will be used. The following formulae govern the specific behavior of these probabilities.

$$(2.2) \quad p_f^n = \begin{cases} 1/3 & 0 \leq n < n_1 \\ 2/3 - 1/3 \cos(\pi(n - n_1)/n_2) & n_1 \leq n < n_1 + n_2 \\ 1 & n_1 + n_2 \leq n < N \end{cases}$$

$$p_t^n = p_w^n = (1 - p_f^n)/2$$

The intervals n_1, n_2 define three distinct regions for these functions. In the first region—which lasts n_1 steps—they describe a uniform distribution. In practice, n_1 should be set fairly large w.r.t. the total number of time steps so that RSAP has plenty of steps to utilize its adaptive features. The second region is a transition region—which lasts for n_2 steps—where proposal fixing begins to dominate the adaptation. From the definitions of p_t^n, p_w^n , there is an equal probability at each step that either thinning or widening will be chosen. There is no firm guideline for setting the second interval except $n_2 \leq N - n_1$, which is logically necessary. With the last region, we forced eventual convergence to $p_f^n \rightarrow 1$ as $n \rightarrow \infty$, and therefore ergodicity (see section 2.3).

Now, the values of σ_t, σ_w adapt based on the number of recent consecutive rejections the chain has encountered, with their default values set to σ_f . If the candidate produced by the method is rejected, we choose from the three kernels based on the defined probability distribution. If the fixed kernel is chosen, then no adaptation is performed. However, if either the thin or wide kernels are used, then we increment one of the counters k_t, k_w and compute the appropriate scale factor

$$(2.3) \quad \begin{aligned} A_t(k_t) &= 1 - (1 - \hat{A}_t)(1 - \exp(-r_t k_t)) \\ A_w(k_w) &= 1 - (1 - \hat{A}_w)(1 - \exp(-r_w k_w)) \end{aligned}$$

where $r_w > 0, r_t > 0, \hat{A}_w > 1$, and $1 > \hat{A}_t > 0$. (These functions—which are identical in form to the probability functions above—were chosen since their initial value is 1, their asymptotic limit is their respective \hat{A} value, and their rate of convergence can be easily controlled via the r constants.) We then set either $\sigma_t = A_t \sigma_f$ or $\sigma_w = A_w \sigma_f$ (depending on which was chosen) and perform a Metropolis step using this proposal width (again remembering that process is

done independently for all parameters). On the other hand, whenever a candidate is accepted, the counters are reset to zero so that $\sigma_t = \sigma_w = \sigma_f$ (for all parameters).

To derive the full M -parameter proposal (which we do in section 2.3 with the help of a great deal of added notation), one has to consider all 3^M permutations of the M parameters selecting the three different types of adaptation. It is helpful to do a two-parameter example. Each parameter will have its own set of proposal widths $\sigma_{t,m} < \sigma_{f,m} < \sigma_{w,m}$ ($m = 1, 2$). Upon adaptation, both parameters must choose which form of adaptation to apply, meaning there are 3^2 possibilities for adaptation. Therefore, defining for the sake of notation $N(\theta_1, \sigma_1^2, \sigma_2^2)$ as a bivariate Gaussian distribution with diagonal covariance matrix given by the diagonal entries (σ_1^2, σ_2^2) , we can write the two-parameter proposal as

$$\begin{aligned} q^n(\theta_{n+1}|\theta_n) &= \sum_{i \in \{t,f,w\}} \sum_{j \in \{t,f,w\}} p_i^n p_j^n N(\theta_n, \sigma_{i,1}^2, \sigma_{j,2}^2) \\ (2.4) \quad &= (p_t^n)^2 N(\theta_n, \sigma_{t,1}^2, \sigma_{t,2}^2) + p_t^n p_f^n N(\theta_n, \sigma_{t,1}^2, \sigma_{f,2}^2) + p_t^n p_w^n N(\theta_n, \sigma_{t,1}^2, \sigma_{w,2}^2) + \\ &\quad p_f^n p_t^n N(\theta_n, \sigma_{f,1}^2, \sigma_{t,2}^2) + (p_f^n)^2 N(\theta_n, \sigma_{f,1}^2, \sigma_{f,2}^2) + p_f^n p_w^n N(\theta_n, \sigma_{f,1}^2, \sigma_{w,2}^2) + \\ &\quad p_w^n p_t^n N(\theta_n, \sigma_{w,1}^2, \sigma_{t,2}^2) + p_w^n p_f^n N(\theta_n, \sigma_{w,1}^2, \sigma_{f,2}^2) + (p_w^n)^2 N(\theta_n, \sigma_{w,1}^2, \sigma_{w,2}^2) \end{aligned}$$

Algorithm 2.1 provides a pseudocode representation of the method. We use a uniform prior over the given domain and a diagonal covariance matrix $\hat{\Sigma}$ for our fixed Gaussian. We define N as the number of time steps and M as the number of parameters. We also define separate counters k_w^m , k_t^m and scale factors A_w^m , A_t^m for every parameter $m = 1, \dots, M$. As can clearly be seen, steps 3-20 are the additional adaptive functionality while steps 21-23 are from the standard Metropolis method.

2.3. Ergodicity of RSAP. x

CHANGES TO MAKE:

Consistent language, notation (hats) (RSAP, inhomogenous MC)

talk about directionality on the graph

We will now provide some analysis of the method, proving conditions a and b of Theorem 5 from Roberts and Rosenthal [8] for our method, thereby showing its ergodicity. Theorem 5 states that an adaptive method is ergodic if it is $a)$ *simultaneously uniformly ergodic* and has $b)$ *diminishing adaptation*. Condition a means that all proposals which could be used in the adaptive method are themselves ergodic if used as proposals for standard Metropolis. Condition b means that the "amount of adaptation" eventually vanishes, i.e., that the sequence of proposals converges (see [8] for more rigorous definitions as well as a proof of Theorem 5). Condition a clearly holds in the case of RSAP since the proposals are all Gaussians, which are ergodic. Therefore, the remainder of this section will be devoted to proving b .

2.3.1. Proving diminishing adaptation. Suppose there are M parameters which RSAP must incorporate and that $\hat{\Sigma}$ is a fixed diagonal covariance matrix for these parameters. First, define the following as the *elemental proposals*

$$(2.5) \quad E((j_1, i_1), \dots, (j_M, i_M)) = N(\theta_1, \Sigma)$$

Algorithm 2.1 Rejection-scaled adaptive proposal (RSAP)

```

1. Initialize  $\theta_0, \hat{\Sigma}, k_w^m = 0, k_t^m = 0$  for all  $m = 1, \dots, M$ 
2. for  $n = 1$  to  $N$  do
3.   if  $n = 1$  or  $\theta'_{n-1}$  was accepted then
4.     Set  $\Sigma = \hat{\Sigma}, k_w^m = 0, k_t^m = 0$  for all  $m$ 
5.   else
6.     for  $m = 1$  to  $M$  do
7.       Choose from {wide, thin, fixed} with probabilities  $\{p_w^n, p_t^n, p_f^n\}$ 
8.       if wide then
9.          $k_w^m = k_w^m + 1$ 
10.         $A_w^m = 1 - (1 - \hat{A}_w)(1 - \exp(-r_w k_w^m))$ 
11.         $\Sigma_{mm} = A_w^m \hat{\Sigma}_{mm}$ 
12.       else if thin then
13.         $k_t^m = k_t^m + 1$ 
14.         $A_t^m = 1 - (1 - \hat{A}_t)(1 - \exp(-r_t k_t^m))$ 
15.         $\Sigma_{mm} = A_{thin}^m \hat{\Sigma}_{mm}$ 
16.       else
17.         $\Sigma_{mm} = \hat{\Sigma}_{mm}$ 
18.       end if
19.     end for
20.   end if
21.   Generate  $\theta'_n \sim N(\theta_{n-1}, \Sigma)$ 
22.   Compute  $\alpha(\theta'_n | \theta_{n-1}) = \min\left(1, \frac{\ell(y|\theta'_n)p(\theta'_n)}{\ell(y|\theta_{n-1})p(\theta_{n-1})}\right)$ 
23.   Set  $\theta_n = \theta'_n$  with probability  $\alpha$ , else  $\theta_n = \theta_{n-1}$ 
24. end for

```

253 where

$$254 \quad (2.6) \quad \Sigma_{mm} = \begin{cases} \hat{\Sigma}_{mm} \cdot A_t(i_m) & j_m = 1 \\ \hat{\Sigma}_{mm} & j_m = 2 \\ \hat{\Sigma}_{mm} \cdot A_w(i_m) & j_m = 3 \end{cases}$$

255 with their respective acceptance probabilities $a_{(j_1, i_1), \dots, (j_M, i_M)}^n = \alpha(\theta'_{n+1} | \theta_n)$ where $m =$
256 $1, \dots, M$ and θ'_{n+1} was drawn from the elemental proposal described by the subscripts. We let n
257 be the time step, the i_m 's be the adaptation degrees, and the j_m 's be switches that determine to
258 which mode of adaptation the i_m 's are applied (1 = thin, 2 = fixed, 3 = wide). Note that n and
259 i_m are non-negative integers such that $n \geq i_m$ since, in practice, the degree of adaptation can-
260 not exceed the number of time steps which have occurred. For the sake of space limitations in
261 this paper, we will also define the rejection probability $r_{(j_1, i_1), \dots, (j_M, i_M)}^n = 1 - a_{(j_1, i_1), \dots, (j_M, i_M)}^n$.

Next, define the following linear combinations as the *adaptive proposals*

$$(2.7) \quad q_{i_1^1, i_1^2, i_1^3, \dots, i_M^1, i_M^2, i_M^3}^n = \left(\sum_{j_1=1}^3 p_{j_1}^n \cdots \sum_{j_M=1}^3 p_{j_M}^n E((j_1, i_1^{j_1}), \dots, (j_M, i_M^{j_M})) \right)$$

with their associated acceptance probabilities

$$(2.8) \quad b_{i_1^1, i_1^2, i_1^3, \dots, i_M^1, i_M^2, i_M^3}^n = \left(\sum_{j_1=1}^3 p_{j_1}^n \cdots \sum_{j_M=1}^3 p_{j_M}^n a_{(j_1, i_1^{j_1}), \dots, (j_M, i_M^{j_M})}^n \right)$$

This is the M -parameter generalization of the one- and two-parameter proposals from section 2.2 with the subscripts t, f, w replaced with 1, 2, 3, respectively. Let's examine Equation 2.5 more closely before continuing on. Regarding the indices, the integers i_m^1 are the degrees of proposal-*thinning* adaptaion while the integers i_m^3 are the degrees of proposal-*widening* adaptaion. The integers i_m^2 can be thought of as the degrees of proposal-*fixing* adaptation (clearly a meaningless notion). They are merely auxiliary indices which help to establish conservation of adaptation degrees. Regarding the sums, there is one per parameter and their indices cover the three adaptation options so that the probability of choosing the elemental proposal $E((j_1, i_1^{j_1}), \dots, (j_M, i_M^{j_M}))$ is $p_{j_1}^n \cdots p_{j_M}^n$.

Lastly, define Q^n as the *full proposal* at time-step n . Since acceptance and rejection are random phenomena, we cannot know ahead of time which specific elemental or adaptive proposal will be used at time step n . Thus, the full proposal will be a linear combination of the adaptive proposals whose total adaptation degrees are less than or equal to n (since there could only have been up to n rejections and, therefore, n adaptations) with functions of their respective acceptance probabilities.

Let us now examine the structure of the full proposal and how it evolves throughout a run, temporarily restricting ourselves to one dimension for simplicity. Below, Figure 1 prodices a graphical representation of the flow of proposals during a run. One can think of method runs as paths through this graph. The first state is always $q'_{0,0,0}$ since no adaptation has yet taken place. Consequently, the first full proposal q^0 is also $q'_{0,0,0}$. As illustrated by the graph, there are four possible transitions from any given state (transitions back to $q'_{0,0,0}$ not shown). If acceptance occurs, the state returns to $q'_{0,0,0}$ (the fixed proposal with no adaptation). Alternatively, if the candidate is rejected, we will increment the adaptive degree indices corrsponding to the elemental proposal which was used. For example, suppose that we chose to use a thin elemental proposal and the sample drawn from it was rejected. Then, we increment the thin proposal index so that the adaptive proposal used in the next time step will be $q'_{1,0,0}$. We can represent this mathematically with linear combinations of the adaptive

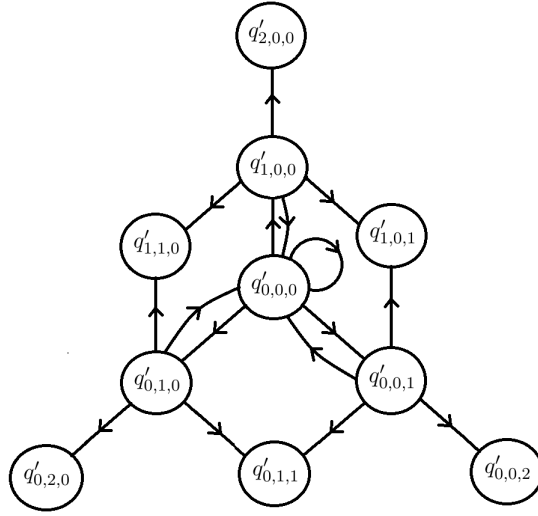


Figure 1. A graphical depiction of the method and the possible transitions that can take place from step one to two. At any state in the graph, four transitions can occur: increasing one of the three indices (rejection) or returning to the initial state.

proposals. Equations 2.8 below show the full proposals through q^2 .

$$\begin{aligned}
 Q^0 &= q_{0,0,0}^0 \\
 Q^1 &= b_{0,0,0}^0 q_{0,0,0}^1 + r_{(1,0)}^0 p_1^0 q_{1,0,0}^1 + r_{(2,0)}^0 p_2^0 q_{0,1,0}^1 + r_{(3,0)}^0 p_3^0 q_{0,0,1}^1 \\
 Q^2 &= \left(b_{0,0,0}^1 b_{0,0,0}^0 + b_{1,0,0}^1 r_{(1,0)}^0 p_1^0 + b_{0,1,0}^1 r_{(2,0)}^0 p_2^0 + b_{0,0,1}^1 r_{(3,0)}^0 p_3^0 \right) q_{0,0,0}^2 + \\
 &\quad r_{(1,0)}^1 p_1^1 b_{0,0,0}^0 q_{1,0,0}^2 + r_{(2,0)}^1 p_2^1 b_{0,0,0}^0 q_{0,1,0}^2 + r_{(3,0)}^1 p_3^1 b_{0,0,0}^0 q_{0,0,1}^2 + \\
 &\quad r_{(1,1)}^1 p_1^1 r_{(1,0)}^0 p_1^0 q_{2,0,0}^2 + r_{(2,1)}^1 p_2^1 r_{(1,0)}^0 p_1^0 q_{1,1,0}^2 + r_{(3,1)}^1 p_3^1 r_{(1,0)}^0 p_1^0 q_{1,0,1}^2 + \\
 &\quad r_{(1,1)}^1 p_1^1 r_{(2,0)}^0 p_2^0 q_{1,1,0}^2 + r_{(2,1)}^1 p_2^1 r_{(2,0)}^0 p_2^0 q_{0,2,0}^2 + r_{(3,1)}^1 p_3^1 r_{(2,0)}^0 p_2^0 q_{0,1,1}^2 + \\
 &\quad r_{(1,1)}^1 p_1^1 r_{(3,0)}^0 p_3^0 q_{1,0,1}^2 + r_{(2,1)}^1 p_2^1 r_{(3,0)}^0 p_3^0 q_{0,1,1}^2 + r_{(3,1)}^1 p_3^1 r_{(3,0)}^0 p_3^0 q_{0,0,2}^2
 \end{aligned}
 \tag{2.9}$$

As can be seen (and was mentioned earlier), Q^n contains all adaptive proposals of total degree up to n , i.e., $i_1^1 + i_1^2 + i_1^3 \leq n$ (for M parameters, this inequality holds for all parameters simultaneously). We can write Q^n succinctly as

$$Q^n = \sum_{\ell=0}^n \left(\sum_{i_1^1 + i_1^2 + i_1^3 = \ell} c_{i_1^1, i_1^2, i_1^3}^n q_{i_1^1, i_1^2, i_1^3}^n \right)
 \tag{2.10}$$

where

$$\sum_{\ell=0}^n \left(\sum_{i_1^1 + i_1^2 + i_1^3 = \ell} c_{i_1^1, i_1^2, i_1^3}^n \right) = 1
 \tag{2.11}$$

defines our normalization (since some proposal must always be chosen) and the inner sums are taken over all possible values of the three indices such that their sum is less than or equal to n . The value $c_{i_1^1, i_1^2, i_1^3}^n$ can be interpreted as the probability that the adaptive proposal defined by the subscripts will be used at step n .

Now, returning to M parameters, let us generalize the above formulae. The full proposal is written

$$(2.12) \quad Q^n = \sum_{\ell=0}^n \left(\sum_{i_1^1 + i_1^2 + i_1^3 = \ell} c_{i_1^1, i_1^2, i_1^3}^n \dots c_{i_M^1, i_M^2, i_M^3}^n q_{i_1^1, i_1^2, i_1^3, \dots, i_M^1, i_M^2, i_M^3}^n \right)$$

with normalization

$$(2.13) \quad \sum_{\ell=0}^n \left(\sum_{i_1^1 + i_1^2 + i_1^3 = \ell} c_{i_1^1, i_1^2, i_1^3}^n \dots c_{i_M^1, i_M^2, i_M^3}^n \right) = 1$$

where $m = 1, \dots, M$. We can construct several recurrence relations which describe how the values of the coefficients c^n change over time. We get one relation which governs the evolution of the coefficient corresponding to the proposal which has zero total adaptation degree and another which governs all the rest.

$$(2.14) \quad \begin{aligned} c_{0,0,0,\dots,0,0,0}^{n+1} &= \sum_{\ell=0}^n \left(\sum_{i_1^1 + i_1^2 + i_1^3 = \ell} b_{i_1^1, i_1^2, i_1^3, \dots, i_M^1, i_M^2, i_M^3}^n c_{i_1^1, i_1^2, i_1^3, \dots, i_M^1, i_M^2, i_M^3}^n \right) \\ c_{i_1^1, i_1^2, i_1^3, \dots, i_M^1, i_M^2, i_M^3}^{n+1} &= \sum_{j_1 | i_1^{j_1} \geq 1} \dots \sum_{j_M | i_M^{j_M} \geq 1} \left(r_{(j_1, i_1^{j_1}), \dots, (j_M, i_M^{j_M})}^n \right) \left(p_{j_1}^n \dots p_{j_M}^n \right) \cdot \\ &\quad c_{i_1^1, \dots, i_1^{j_1-1}, \dots, i_1^3, \dots, i_M^1, \dots, i_M^{j_M-1}, \dots, i_M^3}^n \end{aligned}$$

The first relation is fairly basic, as it simply was derived from the fact that the adaptation-less proposal will be used at any step which immediately follows an acceptance. Consequently, it finds the total, weighted probability of acceptance given all possible proposals which could be used at that step. This is equal to the probability that the adaptation-less proposal will be chosen at the following step. The second relation is more complicated. From a top level perspective, it finds the total, weighted inflowing probability from all proposals which could possibly precede the proposal under consideration. More specifically, it does this by summing the rejection probabilities over all valid permutations of subtracting one from the three types adaptation degrees for all M parameters (of which there are no more than 3^M permutations). By valid, we mean that subtracting one does not give a negative adaptation degree (thus the ≥ 1 under the sums).

Now, to prove diminishing adaptation, we will show that the values of $c_{i_1^1, i_1^2, i_1^3, \dots, i_M^1, i_M^2, i_M^3}^n$ will vanish as $n \rightarrow \infty$ for any proposals that have at least one degree of either thinning or widening adaptation in at least one parameter. We will use induction on the thinning and widening adaptation degrees to prove this.

Let us first prove the base case, where we consider a proposal such that each parameter has only one adaptation degree assigned and there exists at least parameter such that its mode

of adaptation is either thinning or widening. Without loss of generality, we will say that this is the first parameter and that thinning was chosen (the widening case is identical). Letting $c_{1,0,0,\dots,j_M^1,j_M^2,j_M^3}^n$ be the coefficient of this proposal, this implies that either $j_m^1 = 1$ or $j_m^3 = 1$ for every $m = 2, \dots, M$. Then, using the recurrence relation, we get

$$(2.15) \quad c_{1,0,0,\dots,j_M^1,j_M^2,j_M^3}^{n+1} = \left(r_{(1,0),\dots,(j_M,i_M^{j_M})}^n \right) \left(p_1^n \cdots p_{j_M}^n \right) \left(c_{0,0,0,\dots,0,0,0}^n \right)$$

Now, all the terms in this equation are bounded between 0 and 1 and $p_1^n \rightarrow 0$ as $n \rightarrow \infty$. Therefore, $c_{1,0,0,\dots,j_M^1,j_M^2,j_M^3}^{n+1} \rightarrow 0$, as well, and we have that all coefficients with at least one parameter having a single adaptive degree assigned to either thinning or widening vanish.

We will now prove the induction step, which states that if any coefficient with at least one parameter having at least one thinning or widening adaptive degree vanishes, then the coefficients for all proposals that it precedes by one time step (in the sense of the discussion of Equation 2.14) do as well. Consider $c_{i_1^1,i_1^2,i_1^3,\dots,i_M^1,i_M^2,i_M^3}^n$, which has at least one parameter with at least one thinning or widening adaptive degree. Equation 2.14 applied to this coefficient contains the coefficients for all possible preceding states. Now, since there is at least one degree of thinning or widening adaptation in the coefficient under consideration, every term in the sum from Equation 2.14 can be classified in one of two ways: either 1) that adaptation degree was obtained from the immediately preceding step or 2) from an earlier step. If it is from the immediately preceding step, we can factor out the appropriate probability term as we did for the base case, causing that term to vanish. If it was from an earlier step, then the proposal associated with that term already has at least one parameter with at least one thinning or widening adaptive degree, and thus—according to the induction hypothesis—vanishes. Having shown that every term in the sum vanishes, we have shown that all proposal coefficients with at least one parameter with at least one thinning or widening adaptive degree also vanish. Thus, the only non-vanishing coefficients are those for purely fixing proposals, which are all identical to the adaptation-less proposal. Therefore, the full proposal $Q^n \rightarrow q_{0,0,0,\dots,0,0,0}^n$ as $n \rightarrow \infty$, and we have diminishing adaptation, thus proving the ergodicity of RSAP.

3. Computational testing. Having proven the analytical integrity of our method, we now move on to demonstrate its practical utility through the presentation of numerous computational results. These cover multiple different aspects of the method's performance, including flexibility/robustness, mixing efficiency, and rate of convergence. We also present results from a new optimization-style test we developed in which we find the cumulative probability distribution for convergence of the chain to the global minimum of the test function, versus both an interval of initial states (1D only) and a range of fixed proposal widths (1D-MD???). This technique provides easy-to-interpret results which are numerically *and* visually simple to compare.

3.1. Discussion of RSAP parameters values used in testing. When performing numerical experiments on our method, we used standard optimization test functions (including Gaussians, Rosenbrock functions, and Ackley functions), each with a global minimum of $y = f(\theta^*) = 0$ (with the exception of the bi-modal function). As such, the likelihood function

can be expressed as

$$\begin{aligned} \ell(y|\theta) &\propto \exp\left(-\frac{1}{2} \frac{(y - (f(\theta) + \epsilon))^2}{\Delta^2}\right) \\ &\propto \exp\left(-\frac{1}{2} \frac{(f(\theta) + \epsilon)^2}{\Delta^2}\right) \end{aligned}$$

where ϵ is an error term which we include in some of the tests below. The value of Δ affects the ease of acceptance, and thus the relative peakedness of the resulting sample-binned approximation of the posterior. Not knowing its proper value, we include it as a parameter for RSAP to estimate, though we set its proposal width quite low so that it does not change too rapidly, dramatically altering the acceptance rate.

There is also the matter of setting the adaptation parameters to their proper values. Clearly, the optimal values of these parameters are problem-dependent; so we attempted to find a set of reasonable values which improved the performance gain over Metropolis for as many test cases as possible. We thus arrived at maximum scaling magnitudes of $\hat{A}_t = 0.1, \hat{A}_w = 10$, respectively. Being able to both increase and decrease the proposal width by an order of magnitude ensures that the true optimal width is likely contained within the interval $[\hat{A}_t \sigma_f, \hat{A}_w \sigma_f]$. Next, concerning the scaling rates, we found that their values were less significant than the magnitudes in determining performance gain. Their values were set to $r_t = r_w = 0.3$, since this puts the scaling to within approximately 5% of its maximum effect within 10 rejection steps ($e^{-0.3 \cdot 10} \approx 0.049$). Any longer would take too long to reach maximum scaling and frequent acceptances would leave the more extreme scaling values unused. Lastly, the probability scaling intervals n_1, n_2 are likely the most problem-dependent parameters since they determine how long RSAP is allowed to be adaptive. As discussed earlier, $n_1 < N$ should be large w.r.t. N while $n_2 \leq N - n_1$ has more freedom. Because of their problem dependence, we will vary them throughout the tests.

WHY did we use these test functions? Cite Ackley, Rosenbrock, others. Justify usage.

INCLUDE results of modifying adaptation parameters???

UPDATE RSAP Workflow

REDO trace plots and things

3.2. Chain mixing. x

<https://python-graph-gallery.com/111-custom-correlogram/>

To begin our analysis, we will provide trace plots of one-parameter chains created by RSAP for the test functions under consideration, as well as plots of the functions used and sample-binned approximations of the posterior distribution. We will start with the 1D Gaussian defined by

$$f(x) = a \left(1 - \exp\left(-\frac{1}{2} \frac{x^2}{b^2}\right)\right)$$

where $a = 1, b = 0.3, x \in [-1, 1]$. The initial state values were set to $x_0 = 0.8, \Delta_0 = 0.08$ and the fixed proposal widths were set to $\sigma_x = 0.1, \sigma_\Delta = 0.0008$. As discussed above, the parameters for RSAP were set to $\hat{A}_t = 0.1, \hat{A}_w = 10, r_t = 0.3, r_w = 0.3, n_1 = 2000, n_2 = 1000$. The Gaussian is such a simple function that there is no need to use adaptation on it, but it is a good control nonetheless.

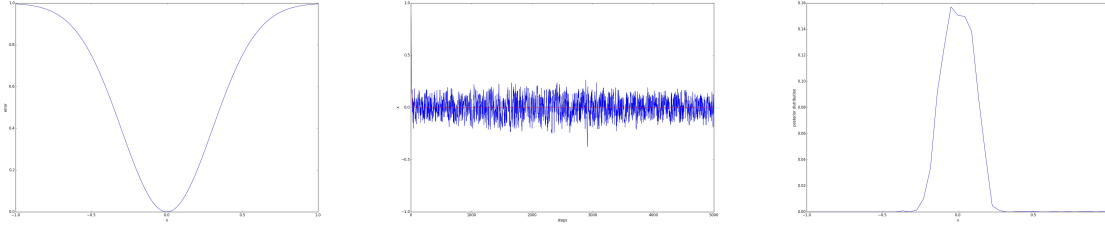


Figure 2. Plots of the Gaussian function, chain trace, and binned posterior estimate resulting from a run with 5,000 steps. In the trace plot in the center, the blue curve is the actual chain and red curve contains the only points with the highest posterior values up to the current time step. For the posterior plot on the right, we used 45 bins to get the image for the posterior distribution.

Next, we have an Ackley function in 1D defined by

$$(3.3) \quad f(x) = a \left(1 - \exp(-0.2\sqrt{x^2/c}) \right) + b \left(e - \exp(\cos(2\pi x)/c) \right)$$

where $a = 20, b = 4, c = 1, x \in [-10, 10]$. The initial state values were set to $x_0 = 8, \Delta_0 = 3.5$ and the fixed proposal widths were set to $\sigma_x = 1.0, \sigma_\Delta = 0.01$. The RSAP parameters again were set to $\hat{A}_t = 0.1, \hat{A}_w = 10, r_t = 0.3, r_w = 0.3, n_1 = 2000, n_2 = 1000$. It is very easy to see

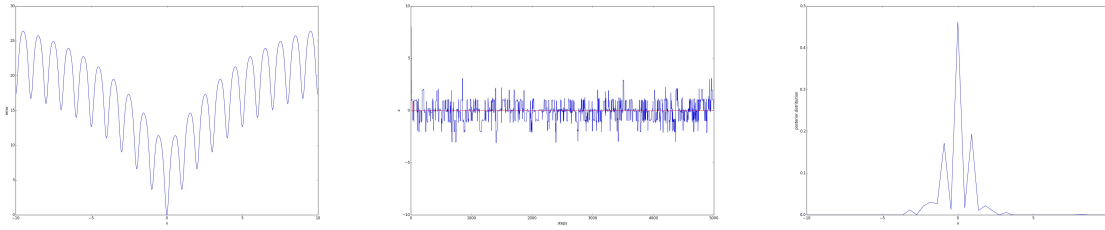


Figure 3. Plots of an Ackley function, chain trace, and binned posterior estimate resulting from a run with 5,000 steps.

from the trace and posterior plots that RSAP efficiently samples the modes near the center.

Lastly, we have a rather illustrative example of the superiority of RSAP to standard Metropolis in the form of a bi-modal function defined by

$$(3.4) \quad f(x) = a \left(1 - \exp\left(-\frac{1}{2} \left(\frac{(x-c)^2}{b^2} \right)^{d/2} \right) \right) + a \left(1 - \exp\left(-\frac{1}{2} \left(\frac{(x+c)^2}{b^2} \right)^{d/2} \right) \right)$$

where $a = 0.5, b = 0.333, c = 0.15, d = 8, x \in [-1, 1]$. The initial state values were set to $x_0 = 0.8, \Delta_0 = 0.08$ and the fixed proposal widths were set to $\sigma_x = 0.1, \sigma_\Delta = 0.0008$. As discussed above, the parameters for RSAP were set to $\hat{A}_t = 0.1, \hat{A}_w = 10, r_t = 0.3, r_w = 0.3, n_1 = 2000, n_2 = 1000$. As can be seen, there is efficient mixing between the two modes in the region up until $n = n_1$. Once standard Metropolis begins to overtake the adaptive effects, the chain becomes stuck in whichever mode it was currently in.

Distributions???

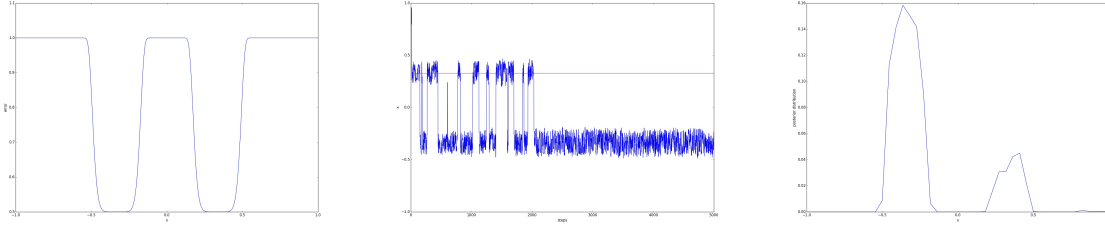


Figure 4. Plots of the bimodal function, chain trace, and binned posterior estimate resulting from an RSAP run with 5,000 steps.

3.3. Convergence to global maximum. We now move on to present results from our optimization-inspired experiments. These used statistics from large ensembles of chains to provide a systematic way of comparing the performance of RSAP and Metropolis over their initial state and their proposal distribution. Comparison over the initial state is useful because it allows us to see if RSAP is actually able to traverse multimodal regions more quickly than Metropolis and by how much. Comparison over proposal width is useful because it allows us to compare both methods in their optimal context, since sweeping through a large range of widths is guaranteed to contain a near-optimal value. As we shall soon see, RSAP consistently outperforms Metropolis for most functions, dimensionalities, and proposal widths (confirm??).

We will begin with convergence versus initial state. For the sake of being able to plot the results, we restrict these experiments to a single dimension.

PLOTS...

Now we will consider convergence versus proposal width. In contrast to the previous experiment, we will present results from tests up to M dimensions. Obviously, since RSAP adapts the proposal width, we will be altering the *fixed* proposal width, letting these values also be the widths for Metropolis. Regarding the chain ensembles used to get the cumulative probability of convergence, we uniformly randomly distribute the initial state of a large number of chains in the (square) search domain. We then mark the time step at which they are within some error tolerance of the global minimum. This threshold changes depending on the scale of the problem so that each function is compared on equal footing.

n_1, n_2 don't matter

RSAP prefers a lower value of Δ than Metropolis, but the optimal value for RSAP is better than the optimal value for Metropolis. This is so that more rejections happen, thus causing RSAP to adapt the proposal.

Might should compare convergence vs. Δ

IMPORTANT: I think since small Δ is good for RSAP but bad for getting a realistic posterior distribution (too peaked), we might should reframe the method as adaptive with smooth burn-in and just prove that the burn-in phase leads to ergodicity. Might have to manually change Δ at the end of the adaptive phase ($n_1 + n_2$)

4. Application: galaxy merger simulation.

5. Conclusions.

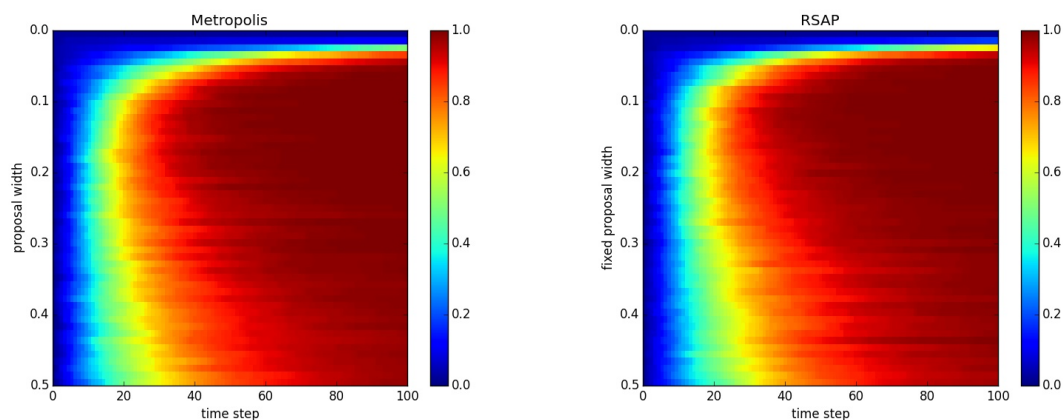


Figure 5. Plots of the cumulative convergence of Metropolis (left) and RSAP (right) on a 1D Gaussian. 100 steps per chain, 500 chains per width value, threshold of 0.003.

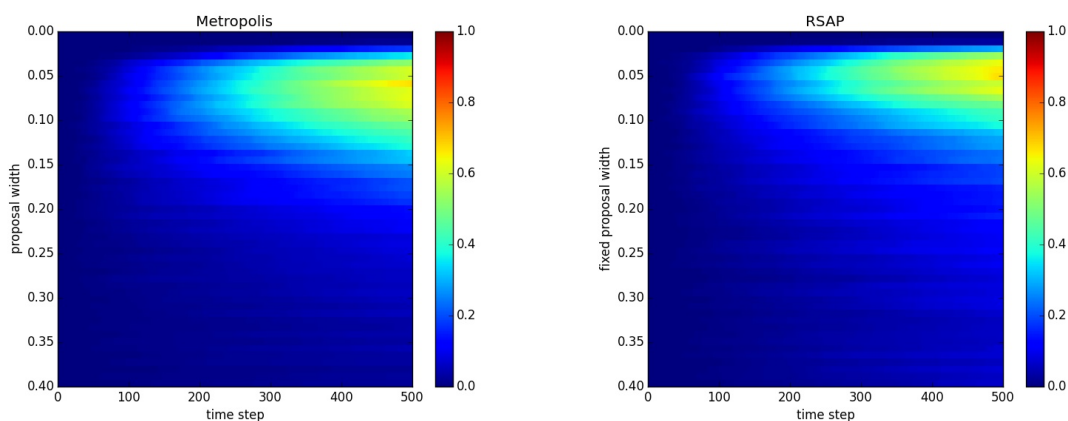


Figure 6. Plots of the cumulative convergence of Metropolis (left) and RSAP (right) on a 3D Gaussian. 500 steps per chain, 1000 chains per width value, threshold of 0.003. Note the different vertical axis scale.

Acknowledgments. We would like to acknowledge the assistance of volunteers in putting together this example manuscript and supplement.

REFERENCES

- [1] Sherlock, Chris; Fearnhead, Paul; Roberts, Gareth O. "The Random Walk Metropolis: Linking Theory and Practice Through a Case Study." *Statist. Sci.* 25 (2010), no. 2, 172-190. doi:10.1214/10-STS327. <https://projecteuclid.org/euclid.ss/1290175840>
- [2] Hastings, W. K. 1970. "Monte Carlo sampling methods using Markov chains and their applications." *Biometrika*, 57:97-109.
- [3] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. 1953. "Equation of state calculations by fast computing machines." *Journal of Chemical Physics*, 21:1087-1092.
- [4] Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., and Rubin, D. *Bayesian Data Analysis*, 3e. 2014. CRC Press.

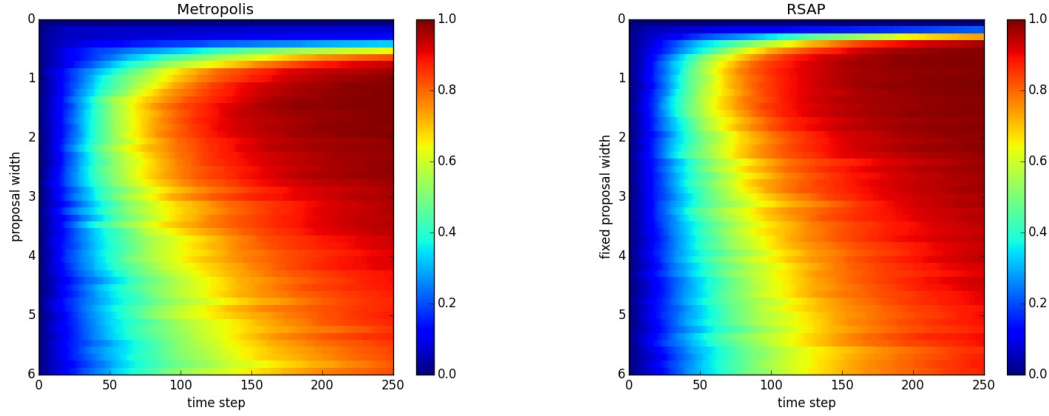


Figure 7. Plots of the cumulative convergence of Metropolis (left) and RSAP (right) on a 1D Ackley function. 150 steps per chain, 1000 chains per width value, threshold of 0.8.

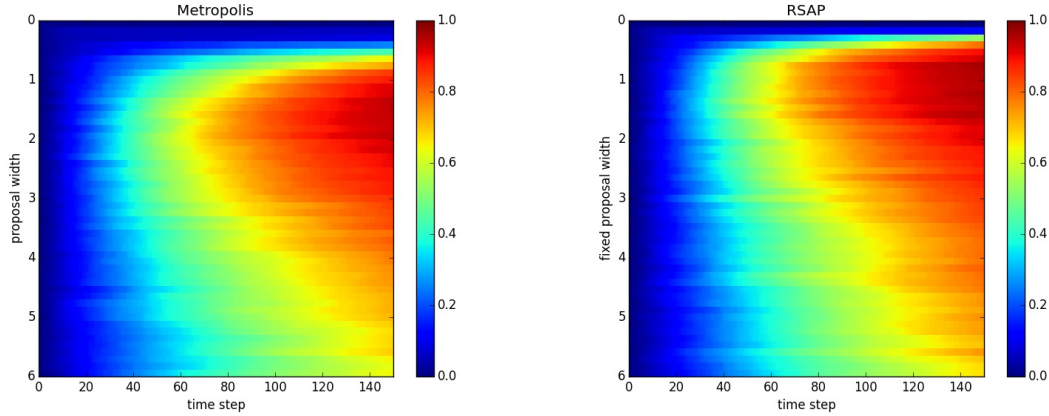


Figure 8. Plots of the cumulative convergence of Metropolis (left) and RSAP (right) on a 1D Ackley function. 150 steps per chain, 1000 chains per width value, threshold of 0.8.

- 472 [5] H. Haario, E. Saksman, and J. Tamminen (1999). Adaptive proposal distribution for random walk
473 Metropolis algorithm. *Comput. Stat.* 14 375-395.
- 474 [6] Harrio, H., Saksman, E., Tamminen, J. "An Adaptive Metropolis Algorithm." *Bernoulli* 7(2), 2001,
475 223-242.
- 476 [7] Gareth O. Roberts and Jeffrey S. Rosenthal (2012) Examples of Adaptive MCMC, *Journal of Computa-*
477 *tional and Graphical Statistics*, 18:2, 349-367, DOI: 10.1198/jcgs.2009.06134
- 478 [8] Gareth O. Roberts and Jeffrey S. Rosenthal. "Coupling and Ergodicity of Adaptive Markov Chain Monte
479 Carlo Algorithms." *Journal of Applied Probability*, Vol. 44, No. 2 (Jun., 2007), pp. 458-475
- 480 [9] Liu Jing, Prahlad Vadakkepat. "Interacting MCMC particle filter for tracking maneuvering target." *Dig-*
481 *ital Signal Processing*. Volume 20, Issue 2, 2010, Pages 561-574.
- 482 [10] A. Holincheck, J. Wallin, K. Borne, L. Fortson, C. Lintott, A M. Smith, S. Bamford, W. Keel, and M.
483 Parrish, "Galaxy Zoo: Mergers - Dynamical Models of Interacting Galaxies." *MNRAS*, (2016). Vol.
484 459, Is. 1, 720-745.
- 485 [11] H. Mo, F. Bosch, and S. White, *Galaxy Formation and Evolution*, 2010, Cambridge University Press.
- 486 [12] J. Wallin, A. Holincheck, and A. Harvey, "JSPAM: A restricted three-body code for simulating interacting

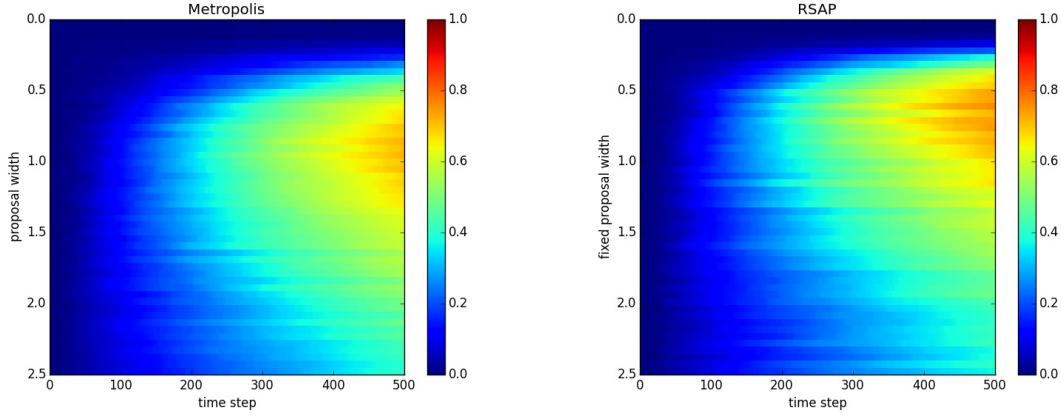


Figure 9. Plots of the cumulative convergence of Metropolis (left) and RSAP (right) on a 2D Ackley function. 500 steps per chain, 1000 chains per width value, threshold of 0.8, $\Delta_0 = 3.5$.

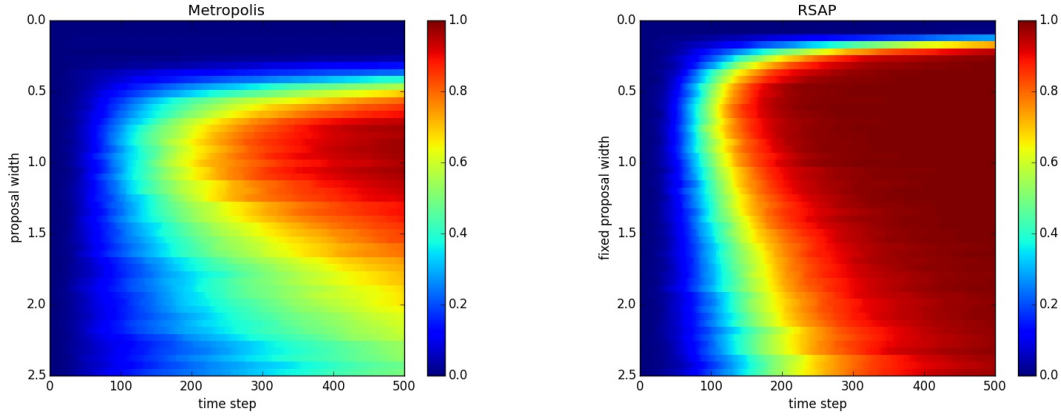


Figure 10. Plots of the cumulative convergence of Metropolis (left) and RSAP (right) on a 2D Ackley function. 500 steps per chain, 1000 chains per width value, threshold of 0.8, $\Delta_0 = 0.5$.

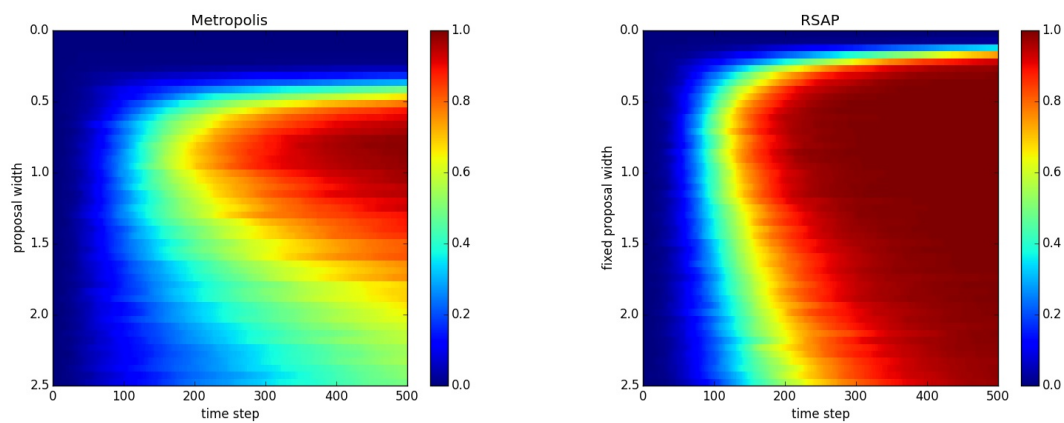


Figure 11. Plots of the cumulative convergence of Metropolis (left) and RSAP (right) on a 2D Ackley function. 500 steps per chain, 1000 chains per width value, threshold of 0.8, $\Delta_0 = 0.01$.