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

Graham West †, John Wallin ‡, and Zachariah Sinkala §

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 inefficiently 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 a new adaptive Metropolis method called 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. Other advantages include not having to locate a near-optimal with a different method beforehand, as well as requiring almost zero computational or storage overhead from standard Metropolis. 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 several 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 and Ackley functions

*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.mtsu.edu).

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

§Department of Computational Science, Middle Tennessee State University, Murfreesboro, TN (zsinkala@mtsu.edu).

as well as a flattened bi-modal function), going into detail in comparing our method's performance with that of standard Metropolis [3] as well as Adaptive Metropolis [6] over a range of proposal widths.

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's Theorem, we have (given data y and model parameters θ)

$$\pi(\theta|y) \propto \ell(y|\theta)p(\theta) \tag{1.1}$$

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.

The Metropolis method has several useful properties which contribute its effectiveness. First—interpreting the method similar to an annealing-type 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,

Algorithm 1.1 The Metropolis method

1: Initialize θ_0
2: **for** $n = 1$ **to** N **do**
3: Generate $\theta'_n \sim q(\theta'_n|\theta_{n-1})$
4: Compute $\alpha(\theta'_n|\theta_{n-1}) = \min\left(1, \dfrac{\pi(\theta'_n|y)}{\pi(\theta_{n-1}|y)}\right)$
$= \min\left(1, \dfrac{\ell(y|\theta'_n)p(\theta'_n)}{\ell(y|\theta_{n-1})p(\theta_{n-1})}\right)$
5: Set $\theta_n = \theta'_n$ with probability α, else $\theta_n = \theta_{n-1}$
6: **end for**

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

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

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

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

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

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

Note that since the stationary distribution—being simply the posterior—is independent of the proposal used, and thus holds for all valid proposals. However, though the method is guaranteed to always converge to π, it may take a large number of steps for convergence to occur. Much of the algorithm's utility rests on the choice of a good proposal q. If one chooses

too thin of a distribution, the method may have difficulty escaping suboptimal modes within an acceptable amount of time. On the other hand, too wide a distribution and the method might jump past true optimal modes or begin to oscillate about them, taking a large number 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 determinstic 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 $P_1P_2P_1P_2\cdots P_k$, 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 $\left(0.5P_1 + 0.5P_2\right)^n$. 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 ver-

ify 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

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

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

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

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.

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

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. The use of the cosine function here is somewhat arbitrary since many other monotonically increasing functions would produce similar results. 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 $n_2 \le N - n_1$, which is logically necessary. With the last region, we forced eventual convergence to $p_f^n \to 1$ as $n \to \infty$, 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

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

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

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

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 $m = 1, ..., M$. 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. Ergodicty of RSAP. x

CHANGES TO MAKE:

Consistent language, notation (hats) (RSAP, inhomogeneous 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*

$$
E((j_1, i_1), \cdots, (j_M, i_M)) = N(\theta_1, \Sigma) \tag{2.5}
$$

Algorithm 2.1 Rejection-scaled adaptive proposal (RSAP)

Initialize θ_0.
for $n = 1$ **to** N **do**
  if $n = 1$ **or** θ'_{n-1} was accepted **then**
    Set $\Sigma = \hat{\Sigma}$, $k_w^m = 0$, $k_t^m = 0$ for all m
  else
    for $m = 1$ **to** M **do**
      Choose from {wide, thin, fixed} with probabilities $\{p_w^n, p_t^n, p_f^n\}$
      if wide **then**
        $k_w^m = k_w^m + 1$
        $A_w^m = 1 - (1 - \hat{A}_w)(1 - \exp(-r_w k_w^m))$
        $\Sigma_{mm} = A_w^m \hat{\Sigma}_{mm}$
      else if thin **then**
        $k_t^m = k_t^m + 1$
        $A_t^m = 1 - (1 - \hat{A}_t)(1 - \exp(-r_t k_t^m))$
        $\Sigma_{mm} = A_{thin}^m \hat{\Sigma}_{mm}$
      else
        $\Sigma_{mm} = \hat{\Sigma}_{mm}$
      end if
    end for
  end if
  Generate $\theta'_n \sim N(\theta_{n-1}, \Sigma)$
  Compute $\alpha(\theta'_n|\theta_{n-1}) = \min\left(1, \dfrac{\ell(y|\theta'_n)p(\theta'_n)}{\ell(y|\theta_{n-1})p(\theta_{n-1})}\right)$
  Set $\theta_n = \theta'_n$ with probability α, else $\theta_n = \theta_{n-1}$
end for

where

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

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

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

$$q^n_{i_1^1,i_1^2,i_1^3,\cdots,i_M^1,i_M^2,i_M^3} = \left(\sum_{j_1=1}^{3} p^n_{j_1} \cdots \sum_{j_M=1}^{3} p^n_{j_M} E((j_1, i_1^{j_1}), \cdots, (j_M, i_M^{j_M}))\right) \tag{2.7}$$

with their associated *acceptance probabilities*

$$b^n_{i^1_1,i^2_1,i^3_1,\cdots,i^1_M,i^2_M,i^3_M} = \left(\sum_{j_1=1}^{3} p^n_{j_1} \cdots \sum_{j_M=1}^{3} p^n_{j_M} a^n_{(j_1,i^{j_1}_1),\cdots,(j_M,i^{j_M}_M)} \right) \tag{2.8}$$

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^1_m are the degrees of proposal-*thinning* adaptation while the integers i^3_m are the degrees of proposal-*widening* adaptation. The integers i^2_m 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 $E((j_1, i^{j_1}_1), \cdots, (j_M, i^{j_M}_M))$ is $p^n_{j_1} \cdots p^n_{j_M}$.

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 provides a graphical representation of the flow of proposals during a run. One can think of

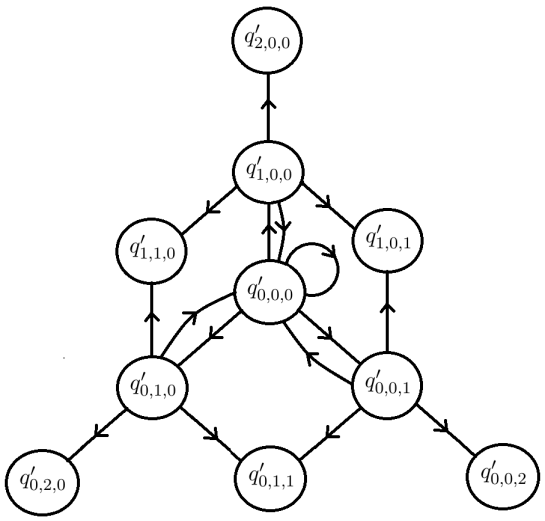

Figure 1. *A graphical depiction of the method and the possible transitions that can take place from step one to two. At any state in the graph, four transitions can occur: increasing one of the three indices (rejection) or returning to the initial state.*

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

$$
\begin{aligned}
Q^0 &= q^0_{0,0,0} \\
Q^1 &= b^0_{0,0,0}q^1_{0,0,0} + r^0_{(1,0)}p^0_1 q^1_{1,0,0} + r^0_{(2,0)}p^0_2 q^1_{0,1,0} + r^0_{(3,0)}p^0_3 q^1_{0,0,1} \\
Q^2 &= \left(b^1_{0,0,0}b^0_{0,0,0} + b^1_{1,0,0}r^0_{(1,0)}p^0_1 + b^1_{0,1,0}r^0_{(2,0)}p^0_2 + b^1_{0,0,1}r^0_{(3,0)}p^0_3\right)q^2_{0,0,0} + \\
&\quad r^1_{(1,0)}p^1_1 b^0_{0,0,0}q^2_{1,0,0} + r^1_{(2,0)}p^1_2 b^0_{0,0,0}q^2_{0,1,0} + r^1_{(3,0)}p^1_3 b^0_{0,0,0}q^2_{0,0,1} + \\
&\quad r^1_{(1,1)}p^1_1 r^0_{(1,0)}p^0_1 q^2_{2,0,0} + r^1_{(2,1)}p^1_2 r^0_{(1,0)}p^0_1 q^2_{1,1,0} + r^1_{(3,1)}p^1_3 r^0_{(1,0)}p^0_1 q^2_{1,0,1} + \\
&\quad r^1_{(1,1)}p^1_1 r^0_{(2,0)}p^0_2 q^2_{1,1,0} + r^1_{(2,1)}p^1_2 r^0_{(2,0)}p^0_2 q^2_{0,2,0} + r^1_{(3,1)}p^1_3 r^0_{(2,0)}p^0_2 q^2_{0,1,1} + \\
&\quad r^1_{(1,1)}p^1_1 r^0_{(3,0)}p^0_3 q^2_{1,0,1} + r^1_{(2,1)}p^1_2 r^0_{(3,0)}p^0_3 q^2_{0,1,1} + r^1_{(3,1)}p^1_3 r^0_{(3,0)}p^0_3 q^2_{0,0,2}
\end{aligned} \tag{2.9}
$$

As can be seen (and was mentioned earlier), Q^n contains all adaptive proposals of total degree up to n, i.e., $i^1_1 + i^2_1 + i^3_1 \leq n$ (for M parameters, this inequality holds for all parameters simultaneously) . We can write Q^n succinctly as

$$
Q^n = \sum_{\ell=0}^{n} \left(\sum_{i^1_1+i^2_1+i^3_1=\ell} c^n_{i^1_1,i^2_1,i^3_1} q^n_{i^1_1,i^2_1,i^3_1} \right) \tag{2.10}
$$

where

$$
\sum_{\ell=0}^{n} \left(\sum_{i^1_1+i^2_1+i^3_1=\ell} c^n_{i^1_1,i^2_1,i^3_1} \right) = 1 \tag{2.11}
$$

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 $c^n_{i^1_1,i^2_1,i^3_1}$ can be interpreted as the probability that 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

$$
Q^n = \sum_{\ell=0}^{n} \left(\sum_{i^1_m+i^2_m+i^3_m=\ell} c^n_{i^1_1,i^2_1,i^3_1,\cdots,i^1_M,i^2_M,i^3_M} q^n_{i^1_1,i^2_1,i^3_1,\cdots,i^1_M,i^2_M,i^3_M} \right) \tag{2.12}
$$

with normalization

$$
\sum_{\ell=0}^{n} \left(\sum_{i^1_m+i^2_m+i^3_m=\ell} c^n_{i^1_1,i^2_1,i^3_1,\cdots,i^1_M,i^2_M,i^3_M} \right) = 1 \tag{2.13}
$$

where $m = 1, \cdots, M$. 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.

$$c^{n+1}_{0,0,0,\cdots,0,0,0} = \sum_{\ell=0}^{n} \sum_{i^1_m + i^2_m + i^3_m = \ell} \left(b^n_{i^1_1,i^2_1,i^3_1,\cdots,i^1_M,i^2_M,i^3_M}\right)\left(c^n_{i^1_1,i^2_1,i^3_1,\cdots,i^1_M,i^2_M,i^3_M}\right)$$

$$c^{n+1}_{i^1_1,i^2_1,i^3_1,\cdots,i^1_M,i^2_M,i^3_M} = \sum_{j_1 | i^{j_1}_1 \geq 1} \cdots \sum_{j_M | i^{j_M}_M \geq 1} \left(r^n_{(j_1,i^{j_1}_1),\cdots,(j_M,i^{j_M}_M)}\right)\left(p^n_{j_1}\cdots p^n_{j_M}\right)\times \left(c^n_{i^1_1,\cdots,i^{j_1}_1-1,\cdots,i^3_1,\cdots,i^1_M,\cdots,i^{j_M}_M-1,\cdots,i^3_M}\right) \tag{2.14}$$

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 acceptance probability 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. Conceptually, it finds the total, weighted rejection probability from all proposals which could possibly precede the proposal under consideration. More specifically, it does this by summing the weighted 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). A permutation is considered valid when subtracting one does not give a negative adaptation degree (thus the $i^{j_m}_m \geq 1$ under the sums).

Now, to prove diminishing adaptation, we will show that the values of $c^n_{i^1_1,i^2_1,i^3_1,\cdots,i^1_M,i^2_M,i^3_M}$ vanish as $n \to \infty$ 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 $c^n_{1,0,0,\cdots,j^1_M,j^2_M,j^3_M}$ be the coefficient of this proposal, this implies that either $j^1_m = 1$ or $j^3_m = 1$ for every $m = 2, \cdots, M$. Then, using the recurrence relation, we get

$$c^{n+1}_{1,0,0,\cdots,j^1_M,j^2_M,j^3_M} = \left(r^n_{(1,0),\cdots,(j_M,i^{j_M}_M)}\right)\left(p^n_1\cdots p^n_{j_M}\right)\left(c^n_{0,0,0,\cdots,0,0,0}\right) \tag{2.15}$$

Now, all the terms in ths equation are bounded between 0 and 1 and $p^n_1 \to 0$ as $n \to \infty$. Therefore, $c^{n+1}_{1,0,0,\cdots,j^1_M,j^2_M,j^3_M} \to 0$, 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 $c^n_{i^1_1,i^2_1,i^3_1,\cdots,i^1_M,i^2_M,i^3_M}$, 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 $Q^n \to q^n_{0,0,0,\cdots,0,0,0}$ as $n \to \infty$, and we have diminishing adaptation, thus proving the ergodicity of RSAP.

3. Computational testing.

Having proven the convergence 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 a range of fixed proposal widths. 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, each with a global minimum of $y = f(\theta^*) = 0$ (the bi-modal function has two adjacent minima). As such, the likelihood function can be expressed as

$$\begin{aligned} \ell(y|\theta) &\propto \exp\left(-\frac{1}{2}\frac{(y-(f(\theta)+\epsilon))^2}{\Delta^2}\right) \\ &\propto \exp\left(-\frac{1}{2}\frac{(f(\theta)+\epsilon)^2}{\Delta^2}\right) \end{aligned} \tag{3.1}$$

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

FIX plots with x as axis label

3.2. Chain mixing. 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

$$f(\theta) = a\Big(1 - \exp(-\frac{1}{2}\frac{\theta^2}{b^2})\Big) \tag{3.2}$$

where $a = 1, b = 0.3, \theta \in [-1, 1]$. The initial state values were set to $\theta_0 = 0.8, \Delta_0 = 0.08$ and the fixed proposal widths were set to $\sigma_\theta = 0.1, \sigma_\Delta = 0.0008$. 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

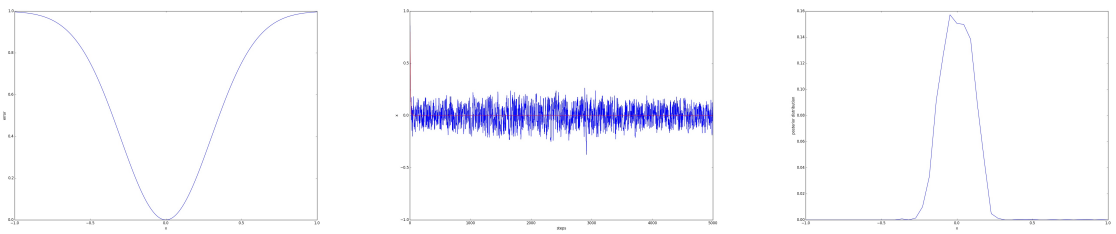

Figure 2. *Plots of the Gaussian function, chain trace, and binned posterior estimate resulting from a run with 5,000 steps. In the trace plot in the center, the blue curve is the actual chain and red curve contains the only points with the highest posterior values up to the current time step. For the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

is a good control nonetheless.

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

$$f(\theta) = a\Big(1 - \exp(-0.2\sqrt{\theta^2/c})\Big) + b\Big(e - \exp(\cos(2\pi\theta)/c)\Big) \tag{3.3}$$

where $a = 20, b = 4, c = 1, \theta \in [-10, 10]$. The initial state values were set to $\theta_0 = 8, \Delta_0 = 3.5$ and the fixed proposal widths were set to $\sigma_\theta = 1.0, \sigma_\Delta = 0.01$. The RSAP parameters again

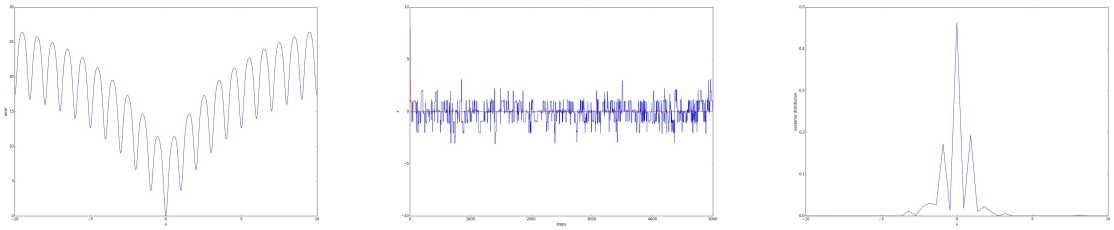

Figure 3. *Plots of an Ackley function, chain trace, and binned posterior estimate resulting from a run with 5,000 steps.*

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 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

$$f(\theta) = a\Big(1 - \exp(-\frac{1}{2}\Big(\frac{(\theta - c)^2}{b^2}\Big)^{d/2})\Big) + a\Big(1 - \exp(-\frac{1}{2}\Big(\frac{(\theta + c)^2}{b^2}\Big)^{d/2})\Big) \tag{3.4}$$

where $a = 0.5, b = 0.15, c = 0.333, d = 8, \theta \in [-1, 1]$. The initial state values were set to $\theta_0 = 0.8, \Delta_0 = 0.08$ and the fixed proposal widths were set to $\sigma_\theta = 0.1, \sigma_\Delta = 0.0008$. 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

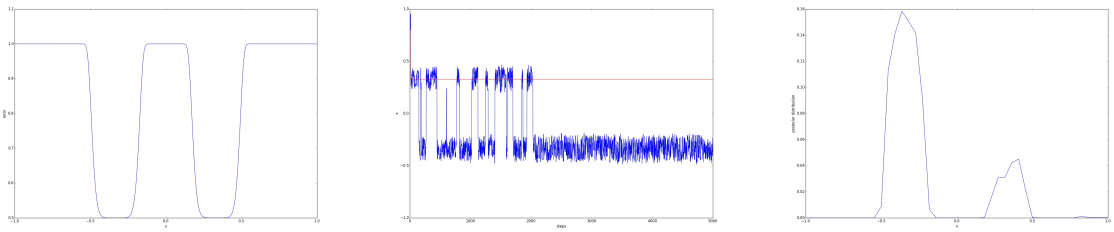

Figure 4. *Plots of the bimodal function, chain trace, and binned posterior estimate resulting from an RSAP run with 5,000 steps.*

the region up until $n = n_1$; however, once standard Metropolis begins to overtake the adaptive effects, the chain becomes temporarily "stuck" in whichever mode it was currently in. In the infinite limit, the mixing between the two modes will equalize, but the use of a small number of steps will result in biased sampling of one mode.

Obviously, this test can be easily skewwed by choosing method parameters which work well for RSAP but poor for Metropolis. The tests in the next section provide a fairer comparison, letting us examine the optimal performance of both methods.

3.3. Convergence to global minimum. We now move on to present results from our optimization-inspired experiments. These use statistics from large ensembles of chains to provide a systematic way of comparing the short-term (i.e., burn-in) performance of RSAP

and Metropolis over different proposal widths. This is useful because it allows us to compare both methods in their optimal context, since sweeping through a large range of proposal widths is guaranteed to contain a near-optimal value. As we shall soon see, RSAP outperforms Metropolis for most functions (only doing Ackley now), dimensionalities, and proposal widths (confirm???).

When performing this test (see Algorithm 3.1), we used M-dimensional Ackley functions

$$f(\theta) = 20\left(1 - \exp\left(-0.2\left(\frac{1}{M}\sum_{m=1}^{M}\theta_m^2\right)^{0.5}\right)\right) + \left(e - \exp\left(\frac{1}{M}\sum_{m=1}^{M}\cos(2\pi\theta_m)\right)\right) \tag{3.5}$$

over a domain defined by the M-dimensional cube $[-L, L]^M$. We sweep through N_{width} proposal widths at equally-spaced intervals, i.e., $\sigma^i = i\sigma_{max}/(N_{width} - 1)$ where $0 \leq i \leq N_{width} - 1$. Obviously, since RSAP adapts the proposal width, we will be sweeping through its *fixed* proposal width, letting these values also equal the widths for Metropolis. We say "widths" rather than "covariance matrices" since—for the Ackley function—each parameter axis is scaled equally, thus causing a single width to work for each. For every width value, we run N_{chains}, each for N_{steps}, with initial states that are uniformly-distributed across the cubical domain. The cumulative probability of convergence to the global minimum of the Ackely function versus the time-step is then calculated for each width by determining at

Algorithm 3.1 Test for convergence to global max

1. Define the M-dimensional parameter vector $\theta^{n,i,j}$ as the n-th time-step of the j-th chain of the i-th proposal width
2. **for** $i = 1$ **to** N_{width} **do**
3. **for** $j = 1$ **to** N_{chain} **do**
4. Initialize chain: $\theta^{0,i,j} \sim U([-L, L]^M)$
5. Calculate $\theta^{0,i,j}$ for $n = 1, ..., N_{step}$ steps using either Metropolis or RSAP with initial state $\theta^{0,i,j}$ and fixed proposal covariance matrix $\Sigma^i = \sigma^i I$
6. **end for**
7. Initialize the cumulative distribution: $c_0^i = 0$
8. **for** $j = 1$ **to** N_{chain} **do**
9. Initialize convergence check: isConv = false
10. **for** $n = 1$ **to** N_{step} **do**
11. **if not** isConv **and** $f(\bar{\theta}_n^{0,i,j}) \leq \varepsilon$ **then**
12. $c_n^i = c_{n-1}^i + 1$
13. isConv = true
14. **else**
15. $c_n^i = c_{n-1}^i$
16. **end if**
17. **end for**
18. **end for**
19. **end for**
20. Normalize c by dividing all entries by N_{chain}

which time-step each chain's Ackley evaluation is below the threshold ε.

Let us note two more considerations before presenting the results. Since we are merely concerned with the short-term behavior of RSAP, the values n_1, n_2 play no role, thus placing us in the adaptive regime the entire time. Also, concerning the value of Δ, lower values are superior for fast initial convergence but poor for getting accurate sampling of the posterior in the long-term. RSAP, in particular, prefers a low value of Δ, since doing so causes more rejections to occur, letting RSAP adapt more frequently. Thus, for the first test, we will include results using several different values of Δ so that its effect can be clearly seen, after which—for the sake of simplicity—we will restrict ourselves to only small values of Δ.

Now, let us examine the results of our numerical tests. Figure 5 contains the results of

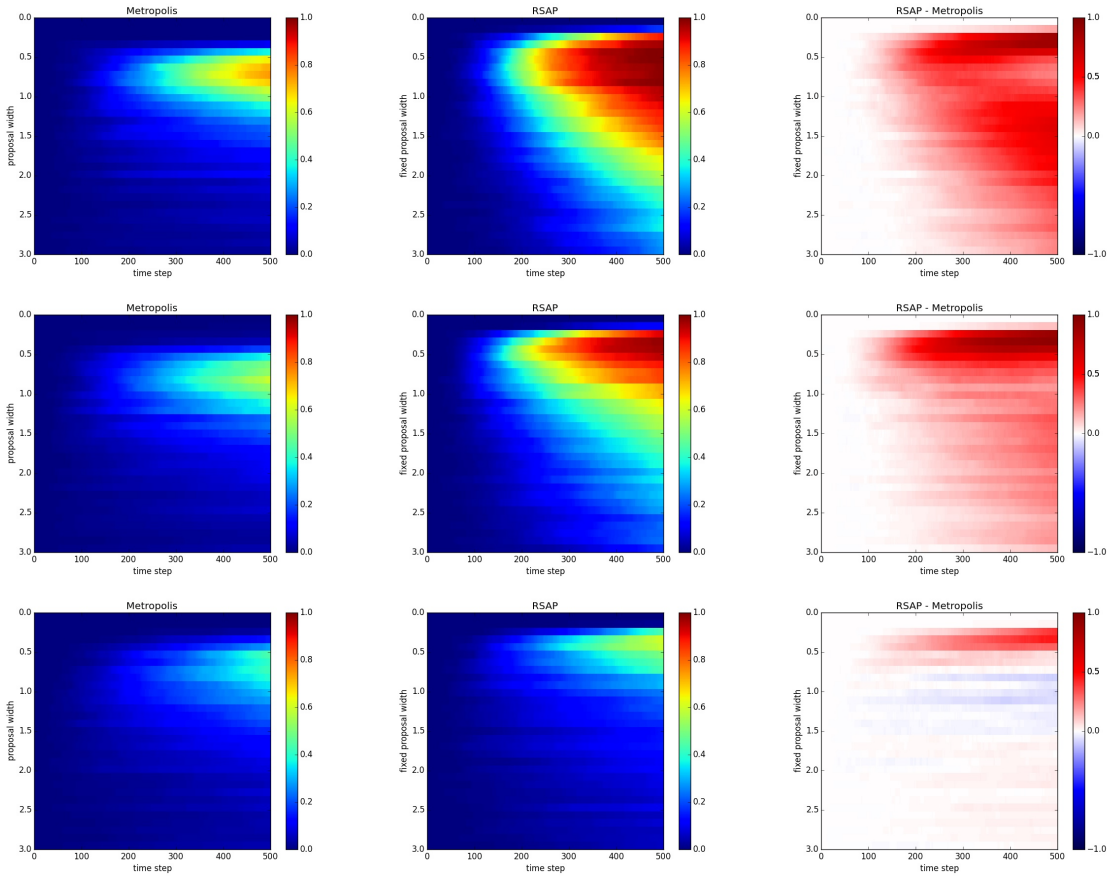

Figure 5. *Plots of the cumulative convergence of Metropolis (left column), RSAP (center column), and their difference (right column) performed on a 3D Ackley function. We used* $\Delta = 0.01$ *(top row),* $\Delta = 1.0$ *(center row), and* $\Delta = 2.0$ *(bottom row). We also used 500 time-steps per chain, 500 chains per width value, and threshold of 1.0 over the domain* $[-15.0, 15.0]^3$.

a comparison between RSAP and Metropolis on a 3D Ackley function. There are several features which must be discussed, most immediately apparent of which is RSAP's superior rate of convergence to the global minimum. Not only does it outperform Metropolis's most

optimal proposal width (roughly 0.6-0.8); it also converges even if it has too small or large of a width for Metropolis to converge at all within the prescribed time frame. Considering just small width values, it is reasonable for Metropolis to stagnate on the Ackley function if given too small a width. Each trough in the Ackley function has a width of 1.0 between adjacent modes. This is due to the $\cos(2\pi\theta_m)$. Now, for a Gaussian distribution, 99.7% of samples drawn will be within 3σ of the mean. Therefore, letting $1.0 = 3\sigma$, we get that a width of 0.333 will only draw samples outside of a particular trough 0.3% of the time (assuming the mean is approximately at the center of the trough, which is reasonable since it will be drawn to that location). Looking back to Figure 5, we see no convergence in Metropolis for width values at or below 0.333. On the other hand, due to RSAP's ability to both grow and shrink the width, it still converges within this region.

It is also clear how significant the effect of changing the value of Δ can be, with a value of 0.01 (top row) performing notably superior to 2.0 (bottom row). For both methods, lower Δ will allow the chains to drift more easily into sub-optimal regions, thus slowing the burn-in process. As stated before, this problem is compounded for RSAP since it depends on rejections to operate effectively.

Next in Figure 6, we have a comparison over a 10D Ackley function. One will notice the

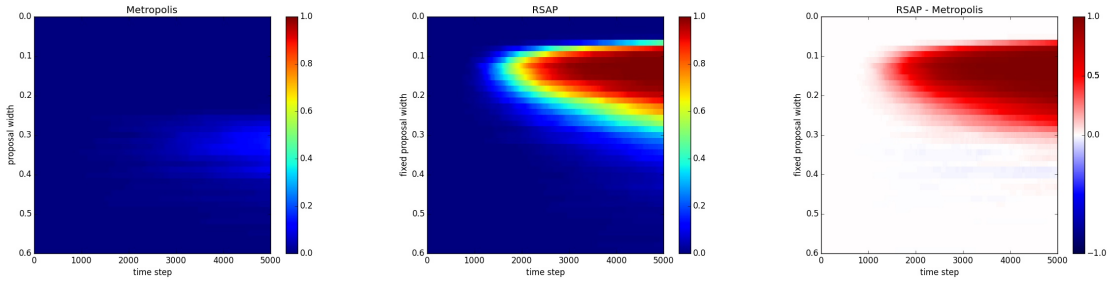

Figure 6. *Plots of the cumulative convergence of Metropolis (left), RSAP (center), and their difference (right) performed on a 10D Ackley function. We used* $\Delta = 0.01$*, 500 time-steps per chain, 500 chains per width value, and threshold of 1.0 over the domain* $[-15.0, 15.0]^1 0$.

scale of the axes has changed dramatically from Figure 5, since a 10D space requires more time-steps to explore and the optimal width decreases [9]. Also, though it was apparent in Figure 5, it is now quite noticable that the optimal (fixed) proposal width for RSAP is less than that for Metropolis. The shift is significant enough that there is a faint blue region in the difference plot (at roughly 0.4), indicating that Metropolis very slightly outperforms RSAP at that width value. On the other hand, looking at the smaller widths, there is no comparison, as RSAP completely dominates Metropolis.

4. Application: galaxy merger simulation.

5. Conclusions.

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.

[5] H. Haario, E. Saksman, and J. Tamminen (1999). Adaptive proposal distribution for random walk Metropolis algorithm. Comput. Stat. 14 375-395.

[6] Harrio, H., Saksman, E., Tamminen, J. "An Adaptive Metropolis Algorithm." *Bernoulli* 7(2), 2001, 223-242.

[7] Gareth O. Roberts and Jeffrey S. Rosenthal (2012) Examples of Adaptive MCMC, Journal of Computational and Graphical Statistics, 18:2, 349-367, DOI: 10.1198/jcgs.2009.06134

[8] Gareth O. Roberts and Jeffrey S. Rosenthal. "Coupling and Ergodicity of Adaptive Markov Chain Monte Carlo Algorithms." *Journal of Applied Probability*, Vol. 44, No. 2 (Jun., 2007), pp. 458-475

[9] G. O. Roberts, A. Gelman, and W. R. Gilks (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms. Ann. Appl. Probab. 7, 110-20.

[10] Liu Jing, Prahlad Vadakkepat. "Interacting MCMC particle filter for tracking maneuvering target." *Digital Signal Processing.* Volume 20, Issue 2, 2010, Pages 561-574.

[11] A. Holincheck, J. Wallin, K. Borne, L. Fortson, C. Lintott, A M. Smith, S. Bamford, W. Keel, and M. Parrish, "Galaxy Zoo: Mergers - Dynamical Models of Interacting Galaxies." *MNRAS*, (2016). Vol. 459, Is. 1, 720-745.

[12] H. Mo, F. Bosch, and S. White, *Galaxy Formation and Evolution*, 2010, Cambridge University Press.

[13] J. Wallin, A. Holincheck, and A. Harvey, "JSPAM: A restricted three-body code for simulating interacting galaxies." *Astronomy and Computing* 16, (2016). 26-33.