Examples of Adaptive MCMC

by

Gareth O. Roberts[*] and Jeffrey S. Rosenthal[**]

(September 2006; revised January 2008.)

Abstract. We investigate the use of adaptive MCMC algorithms to automatically tune the Markov chain parameters during a run. Examples include the Adaptive Metropolis (AM) multivariate algorithm of Haario et al. (2001), Metropolis-within-Gibbs algorithms for non-conjugate hierarchical models, regionally adjusted Metropolis algorithms, and logarithmic scalings. Computer simulations indicate that the algorithms perform very well compared to non-adaptive algorithms, even in high dimension.

1. Introduction.

MCMC algorithms such as the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970) are extremely widely used in statistical inference, to sample from complicated high-dimensional distributions. Tuning of associated parameters such as proposal variances is crucial to achieve efficient mixing, but can also be very difficult.

Adaptive MCMC algorithms attempt to deal with this problem by automatically "learning" better parameter values of Markov chain Monte Carlo algorithms while they run. In this paper, we consider a number of examples of such algorithms, including some in high dimensions. We shall see that adaptive MCMC can be very successful at finding *good* parameter values with little user intervention. In our context, good will be defined in terms of some appropriate measure of Markov chain mixing, such as the integrated autocorrelation of a functional of interest.

It is known that adaptive MCMC algorithms will not always preserve stationarity of $\pi(\cdot)$, see e.g. Rosenthal (2004) and Proposition 3 of Roberts and Rosenthal (2005). However, they will converge if the adaptions are done at regeneration times (Gilks et al., 1998; Brockwell and Kadane, 2005), or under various technical conditions about the adaption procedure

[*]Department of Mathematics and Statistics, Fylde College, Lancaster University, Lancaster, LA1 4YF, England. Email: `g.o.roberts@lancaster.ac.uk`.

[**]Department of Statistics, University of Toronto, Toronto, Ontario, Canada M5S 3G3. Email: `jeff@math.toronto.edu`. Web: `http://probability.ca/jeff/` Supported in part by NSERC of Canada.

(Haario et al., 2001; Atchadé and Rosenthal, 2005; Andrieu and Moulines, 2003; Andrieu and Atchadé, 2006).

Roberts and Rosenthal (2005) proved ergodicity of adaptive MCMC under conditions which we find simpler to apply, and which do not require that the adaptive parameters converge. To state their result precisely, suppose the algorithm updates X_n to X_{n+1} using the kernel P_{Γ_n}, where each fixed kernel P_γ has stationary distribution $\pi(\cdot)$, but where the Γ_n are random indices, chosen iteratively from some collection $\mathcal{Y}$ based on past algorithm output. Write $\|\cdots\|$ for total variation distance, $\mathcal{X}$ for the state space, and $M_\epsilon(x,\gamma) = \inf\{n \geq 1 : \|P_\gamma^n(x,\cdot) - \pi(\cdot)\| \leq \epsilon\}$ for the convergence time of the kernel P_γ when beginning in state $x \in \mathcal{X}$. Then Theorem 13 of Roberts and Rosenthal (2005), combined slightly with their Corollaries 8 and 9 and Theorem 23 guarantee that $\lim_{n\to\infty} \|\mathcal{L}(X_n) - \pi(\cdot)\| = 0$ (asymptotic convergence), and also $\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^n g(X_i) = \pi(g)$ for all bounded $g : \mathcal{X} \to \mathbf{R}$ (WLLN), assuming only the *Diminishing Adaptation* condition

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

and the *Bounded Convergence* condition

$$\{M_\epsilon(X_n,\Gamma_n)\}_{n=0}^\infty \quad \text{is bounded in probability}\,, \qquad \epsilon > 0\,. \tag{2}$$

Furthermore, they prove that (2) is satisfied whenever $\mathcal{X} \times \mathcal{Y}$ is finite, or is compact in some topology in which either the transition kernels P_γ, or the Metropolis-Hastings proposal kernels Q_γ, have jointly continuous densities. (Condition (1) can be ensured directly, by appropriate design of the adaptive algorithm.) A SLLN is precluded since the convergence statements above are only stated "in probability", while CLTs do not necessarily hold since Γ_n does not necessarily converge at all.

Such results provide a "hunting license" to look for useful adaptive MCMC algorithms. In this paper, we shall consider a variety of such algorithms. We shall see that they do indeed converge correctly, and often have significantly better mixing properties than comparable non-adaptive algorithms.

We present a collection of examples. For each one, our adaptive strategy steers the algorithm towards a desired operational "optimal" according to some prescribed criterion. Crucially, our approach differs from that of Andrieu and Moulines, 2003 and Andrieu and Atchadé, in that unlike our method, convergence of the adaptive strategy is specifically sought in their approach. Our regularity conditions are thus weaker and easier to verify, though as a result, the results we can demonstrate are necessarily weaker also.

2. Adaptive Metropolis (AM).

In this section, we consider a version of the Adaptive Metropolis (AM) algorithm of Haario et al. (2001). We begin with a d-dimensional target distribution $\pi(\cdot)$. We perform a Metropolis algorithm with proposal distribution given at iteration n by $Q_n(x,\cdot) = N(x,\ (0.1)^2\, I_d\,/\,d)$ for $n \le 2d$, while for $n > 2d$,

$$Q_n(x,\cdot) \;=\; (1-\beta)\, N(x,\, (2.38)^2\, \Sigma_n\,/\,d) \;+\; \beta\, N(x,\ (0.1)^2\, I_d\,/\,d)\,, \tag{3}$$

where Σ_n is the current empirical estimate of the covariance structure of the target distribution based on the run so far, and where β is a small positive constant (we take $\beta = 0.05$).

It is known from Roberts et al. (1997) and Roberts and Rosenthal (2001) that the proposal $N(x,\, (2.38)^2\, \Sigma\,/\,d)$ is optimal in a particular large-dimensional context. Thus, the $N(x,\, (2.38)^2\, \Sigma_n\,/\,d)$ proposal is an effort to approximate this.

Since empirical estimates change at the n^{th} iteration by only $O(1/n)$, it follows that (1) will be satisfied. Restricting $\beta > 0$ in (3) ensures that (2) is satisfied, at least for a large family of target densities which includes all those which are log-concave outside some arbitrary bounded region (see Section 8). Hence, this algorithm will indeed converge to $\pi(\cdot)$ and satisfy the WLLN. (Haario et al. instead let $Q_n(x,\cdot) = N(x,\, \Sigma_n + \epsilon\, I_d)$ for small ϵ, to force $c_1 I_d \le \Sigma_n \le c_2 I_d$ for some $c_1, c_2 > 0$, which also ensures (1) and (2) for target distributions with bounded support, but we prefer to avoid this strong assumption.)

To test this algorithm, we let $\pi(\cdot) = N(0,\ M\, M^t)$, where M is a $d \times d$ matrix generated randomly by letting $\{M_{ij}\}_{i,j=1}^{d}$ be i.i.d. $\sim N(0,1)$. This ensures that the target covariance matrix $\Sigma = M\, M^t$ will be highly erratic, so that sampling from $\pi(\cdot)$ presents a significant challenge for sampling if the dimension is at all high.

The resulting trace plot of the first coordinate of the Markov chain is presented in Figure 1 for dimension $d = 100$, and in Figure 2 for dimension $d = 200$. In both cases, the Markov chain takes a long time to adapt properly and settle down to rapid mixing. In the early stages, the algorithm vastly underestimates the true stationary variance, thus illustrating the pitfalls of premature diagnoses of MCMC convergence. In the later stages, by contrast, the algorithm has "learned" how to sample from $\pi(\cdot)$, and does so much more successfully.

Another way of monitoring the success of this algorithm's adapting is as follows. Consider a multi-dimensional random-walk Metropolis algorithm with proposal covariance matrix $(2.38)^2\, \Sigma_p\,/\,d$, acting on a normal target distribution with true covariance matrix Σ. Theorem 5 of Roberts and Rosenthal (2001) prove that it is optimal to take $\Sigma_p = \Sigma$, and for

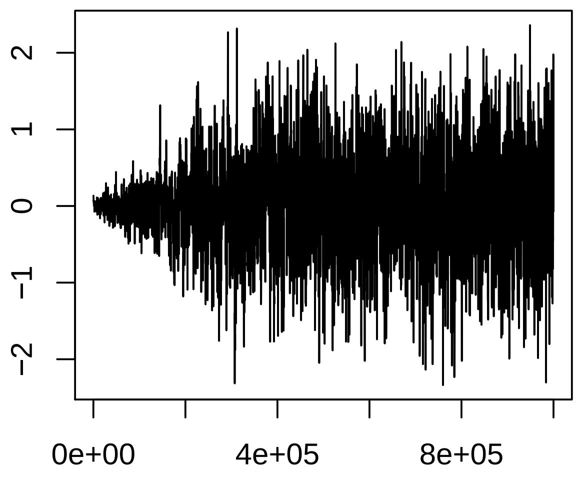

Figure 1. The first coordinate of the AM Markov chain in dimension 100, plotted against iteration number.

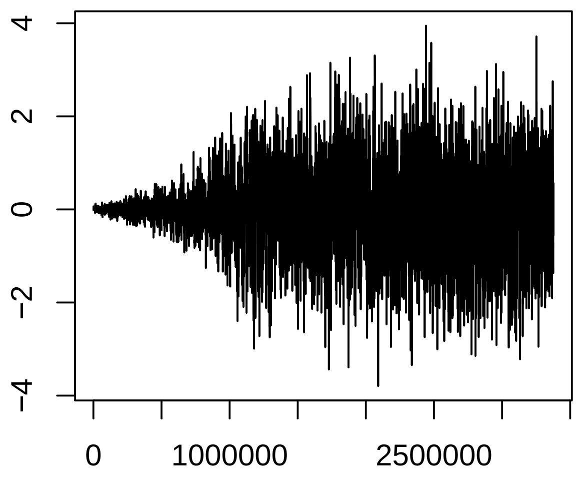

Figure 2. The first coordinate of the AM Markov chain in dimension 200, plotted against iteration number.

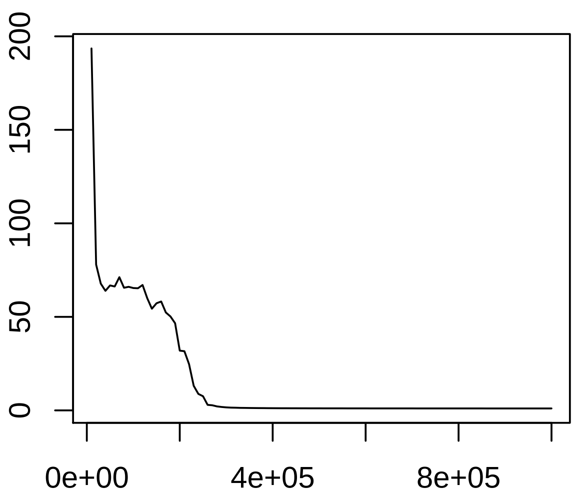

Figure 3. The suboptimality factor b for the AM algorithm in dimension 100, plotted against iteration number.

other Σ_p the mixing rate will be slower than this by a sub-optimality factor of

$$b \;\equiv\; d\,\frac{\sum_{i=1}^{d}\lambda_i^{-2}}{(\sum_{i=1}^{d}\lambda_i^{-1})^2}\,,$$

where $\{\lambda_i\}$ are the eigenvalues of the matrix $\Sigma_p^{1/2}\,\Sigma^{-1/2}$. Usually we will have $b > 1$, and the closer b is to 1, the better. The criterion being optimised in AM is therefore b^{-1}.

So how does the AM algorithm perform by this measure? For the run in dimension 100, the value of this sub-optimality coefficient b begins at the huge value of 193.53, and then eventually decreases towards 1, reaching 1.086 after 500,000 iterations, and 1.024 after 1,000,000 iterations (Figure 3). In dimension 200, the value of b is even more erratic, starting around 183,000 and oscillating wildly before decreasing to about 1.04 after 800000 iterations.

We conclude from this that the AM algorithm does indeed "learn" about the true target covariance matrix, and converge to an algorithm which samples very (almost optimally) efficiently from $\pi(\cdot)$. It is true that it takes many iterations for the algorithm to learn this information (nearly 400,000 iterations in dimension 100, and nearly 2,000,000 in dimension 200). On the other hand, what the algorithm is learning is a $d\times d$ covariance matrix with many parameters (5,050 parameters in dimension 100, and 20,100 in dimension 200). We feel that this indicates very impressive performance of the AM algorithm in high dimensions.

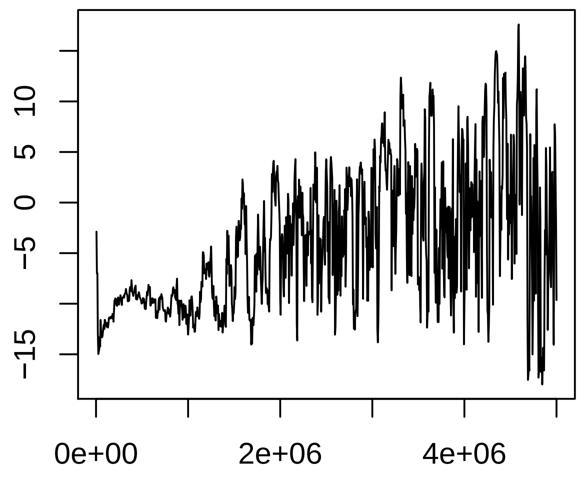

Figure 4. Trace plot of the first coordinate in the banana-shaped example.

2.1. An irregularly shaped example.

AM can be expected to work well on target densities in which the density contours form roughly elliptical contours. In such examples the global covariance gives a good measure of dependence valid in all parts of the state space. However, it is interesting to see how the approach performs on a more challenging problem with more irregularly shaped contours.

We also applied our full Adaptive Metropolis algorithm to a "banana-shaped" distribution, as proposed by Haario et al. (1999, 2001), with density

$$f_B = f_d \circ \phi_B$$

where f_d is the d-dimensional density of a $N(\mathbf{0},\ \mathrm{diag}(100, 1, 1, \ldots, 1))$ distribution, and where $\phi_B(x_1, \ldots, x_d) = (x_1, x_2 + Bx_1^2 - 100B, x_3, \ldots, x_d)$ with $B > 0$ the "bananicity" constant. So,

$$f_B(x_1, \ldots, x_d) \ \propto\ \exp\Big[- x_1^2/200 - \frac{1}{2}(x_2 + Bx_1^2 - 100B)^2 - \frac{1}{2}(x_3^2 + x_4^2 + \ldots + x_d^2)\Big]\,.$$

Specifically, we take dimension $d = 20$, and take $B = 0.1$, and run the algorithm for 5,000,000 iterations. A trace plot of the first coordinate is given in Figure 4.

It is clear that the adaptation has improved mixing here. However mixing is still very poor after 5000000 iterations which is to be expected given that ant Metropolis method struggles to traverse this distributions support. Whilst the AM algorithm attempts to move around as effectively as it can, classes of algorithms which can adjust the covariance of the proposal distribution according to the current state of the algorithm should be required. This in part motivates some of the methods we shall introduce in later sections.

3. Adaptive Metropolis-Within-Gibbs.

Consider the following statistical model:

$$\begin{tikzcd}
& \mu \arrow[dl] \arrow[d] \arrow[dr] & & & \\
\theta_1 \arrow[d] & \cdots \quad \cdots \arrow[d] & \theta_K \arrow[d] & \theta_i \sim \text{Cauchy}(\mu, A) & [1 \le i \le K] \\
Y_{11}, \dots, Y_{1r_1} & & Y_{K1}, \dots, Y_{Kr_K} & Y_{ij} \sim N(\theta_i, V) & [1 \le j \le r_i]
\end{tikzcd}$$

with priors $N(0,1)$ on μ, and $IG(1,1)$ on A and V. Here $\{Y_{ij}\}$ are observed data, $IG(a,b)$ is the inverse gamma distribution with density proportional to $e^{-b/x}x^{-(a+1)}$, and Cauchy(m,s) is a translated and scaled Cauchy distribution with density proportional to $[1 + ((x - m)/s)^2]^{-1}$. This model gives rise to a posterior distribution $\pi(\cdot)$ on the $(K+3)$-dimensional vector $(A, V, \mu, \theta_1, \dots, \theta_K)$, conditional on the observed data $\{Y_{ij}\}$.

We take $K = 500$, and let the r_i vary between 5 and 500. The resulting model is too complicated for analytic computation, and far too high-dimensional for numerical integration. Furthermore, the presence of the Cauchy (as opposed to Normal) distribution destroys conjugacy, and thus makes a classical Gibbs sampler (as in Gelfand and Smith, 1990) infeasible. Instead, a Metropolis-within-Gibbs algorithm (Metropolis et al., 1953; Tierney, 1994) seems appropriate.

Such an algorithm might proceed as follows. We consider each of the 503 variables in turn. For each, we propose updating its value by adding a $N(0, \sigma^2)$ increment. That proposal is then accepted or rejected according to the usual Metropolis ratio. This process is repeated many times, allowing the variables to hopefully converge in distribution to $\pi(\cdot)$. But how should σ^2 be chosen? Should it be different for different variables? How can we feasibly determine appropriate scalings in such high dimension?

To answer these questions, an adaptive algorithm can be used. We proceed as follows. For each of the variables i $[1 \le i \le K+3]$, we create an associated variable ls_i giving the logarithm of the standard deviation to be used when proposing a normal increment to variable i. We begin with $ls_i = 0$ for all i (corresponding to unit proposal variance). After the n^{th} "batch" of 50 iterations, we update each ls_i by adding or subtracting an adaption amount $\delta(n)$. The adapting attempts to make the acceptance rate of proposals for variable i

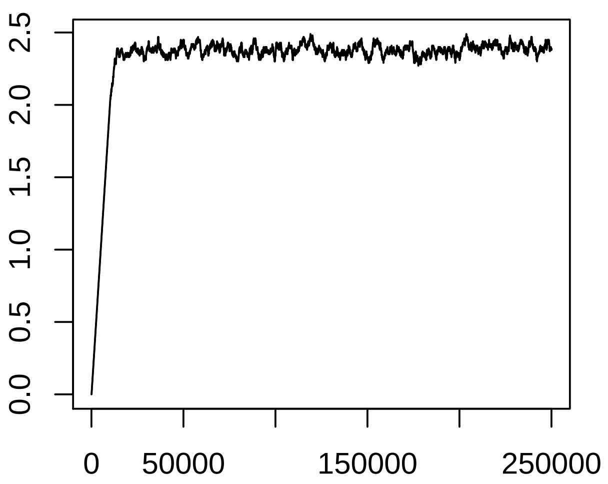

Figure 5. The log proposal standard deviation ls_1 corresponding to the Metropolis-within-Gibbs variable θ_1, plotted against batch number.

as close as possible to 0.44 (which is optimal for one-dimensional proposals in certain settings, cf. Roberts et al., 1997; Roberts and Rosenthal, 2001). Specifically, we increase ls_i by $\delta(n)$ if the fraction of acceptances of variable i was more than 0.44 on the n^{th} batch, or decrease ls_i by $\delta(n)$ if it was less.

Condition (1) is satisfied provided $\delta(n) \to 0$; we take $\delta(n) = \min(0.01, n^{-1/2})$. Our approach is to specify a global maximal parameter value $M < \infty$, and restrict each ls_i to the interval $[-M, M]$. For a large class of target densities (which includes all those which are log-concave outside an arbitrary bounded region) this ensures (2) hold. In practice, the ls_i stabilise nicely so the bound on M is not actually needed.

To test this adaptive algorithm, we generate independent test data $Y_{ij} \sim N(i-1,\ 10^2)$, for $1 \le i \le 500$ and $1 \le j \le r_i$. For such data, our simulations show that the scaling variables quickly settle down near "good" values where acceptance rates are roughly 0.44. Indeed, for the location variables θ_1, θ_2, and θ_3, the corresponding ls variables converge to values near 2.4, 1.2, and 0.1, respectively (Figures 5, 6, 7). So the algorithm appears to be converging well.

Just how good are the values chosen? The following table presents the integrated autocorrelation times (ACT) and average squared jumping distances (after discarding the first fifth of the run as burn-in), for both the adaptive algorithm, and the corresponding "fixed"

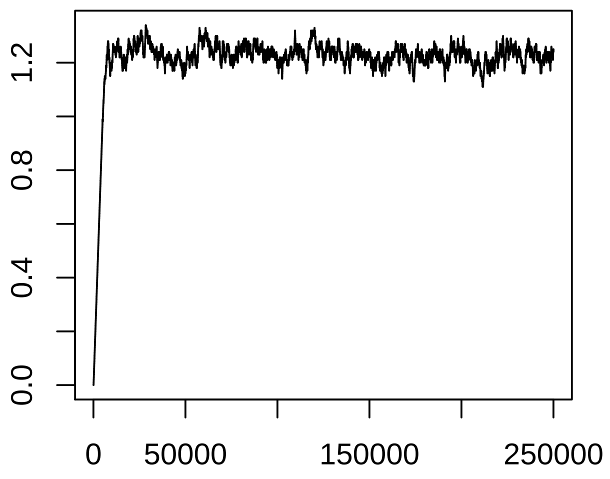

Figure 6. **The log proposal standard deviation ls_2 corresponding to the Metropolis-within-Gibbs variable θ_2, plotted against batch number.**

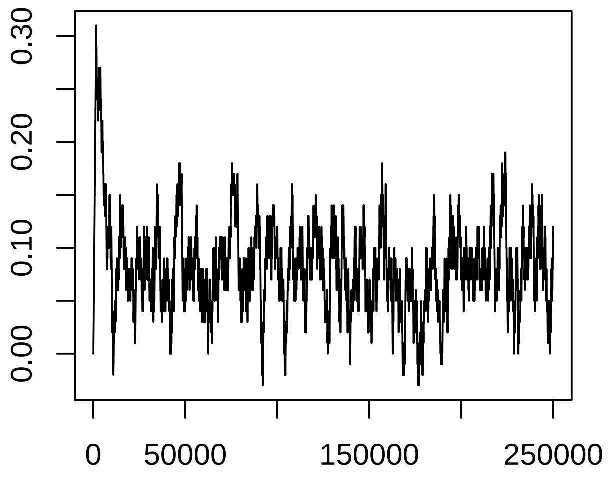

Figure 7. **The log proposal standard deviation ls_3 corresponding to the Metropolis-within-Gibbs variable θ_3, plotted against batch number.**

algorithm where each ls_i is fixed at 0:

Variable	r_i	Algorithm	ACT	Avr Sq Dist
θ_1	5	Adaptive	2.59	14.932
θ_1	5	Fixed	31.69	0.863
θ_2	50	Adaptive	2.72	1.508
θ_2	50	Fixed	7.33	0.581
θ_3	500	Adaptive	2.72	0.150
θ_3	500	Fixed	2.67	0.147

This table shows that, when comparing adaptive to fixed algorithms, for variables θ_1 and θ_2, the autocorrelation times are significantly smaller (better) and the average squared jumping distances are significantly larger (better), Thus, adapting has significantly improved the MCMC algorithm, by automatically choosing appropriate proposal scalings separately for each coordinate. For variable θ_3 the performance of the two algorithms is virtually identical, which is not surprising since (Figure 7) the optimal log proposal standard deviation happens to be very close to 0 in that case.

In summary, this adaptive algorithm appears to correctly scale the proposal standard deviations, leading to a Metropolis-within-Gibbs algorithm which mixes much faster than a naive one with unit proposal scalings. Coordinates are improved wherever possible, and are left about the same when they happen to already be optimal. This works even in high dimensions, and does not require any direct user intervention or high-dimensional insight. This algorithm has recently been applied to a statistical genetics problem (Turro et al., 2007).

3.1. A comparison with SCAM.

A different component-wise adaptive scaling method, the Single Component Adaptive Metropolis (SCAM) algorithm, is presented in Haario et al. (2005). That algorithm, which resembles the Adaptive Metropolis algorithm of Haario et al. (2001), is very interesting and promising, but differs significantly from ours since the SCAM adapting is done based on the empirical variance of each component based on the run so far.

For comparative purposes, we also ran the SCAM algorithm of Haario et al. (2005) on the same example as that above for adaptive Metropolis-within-Gibbs. The SCAM algorithm uses the proposal distribution $Y_n^i \sim N(X_{n-1}^i, v_n^i)$ for the i^{th} coordinate, where

$$v_n^i = \begin{cases} 5^2, & n \le 10 \\ (2.4)^2(g_n^i + 0.05), & n \ge 11 \end{cases}$$

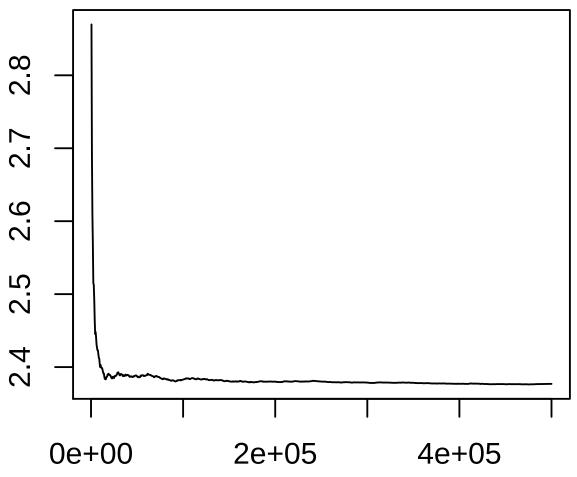

Figure 8. The log proposal standard deviation ls_1 corresponding to the SCAM variable θ_1, plotted against iteration number.

Here g_n^i is the sample variance of $X_0^{(i)}, X_1^{(i)}, \ldots, X_{n-1}^{(i)}$. (Intuitively, for $n \geq 11$, v_n^i attempts to mimic an "optimal" one-dimensional variance $(2.38)^2 \operatorname{Var}_\pi(X_i)$ similar to what was discussed above; the published version of SCAM omits the square in "2.4" but we assume the above is what was intended.) Writing $\overline{x}_n^i = \frac{1}{n}\sum_{j=0}^{n-1} x_j^{(i)}$, we see (cf. Haario et al., 2005) that we can use the recursive equations

$$\overline{x}_n^i \;=\; \frac{n-1}{n}\overline{x}_{n-1}^i + \frac{1}{n}x_{n-1}^i$$

and

$$g_n^i \;=\; \frac{n-2}{n-1}g_{n-1}^i + (\overline{x}_{n-1}^i)^2 + \frac{1}{n-1}(x_n^i)^2 - \frac{n}{n-1}(\overline{x}_n^i)^2\,.$$

We again consider the first three coordinates, as above. The graphs of their proposal variances (again on a log scale, for consistency with the above) are presented here.

We also compute the mean log σ, ACT, and average squared jumping distance:

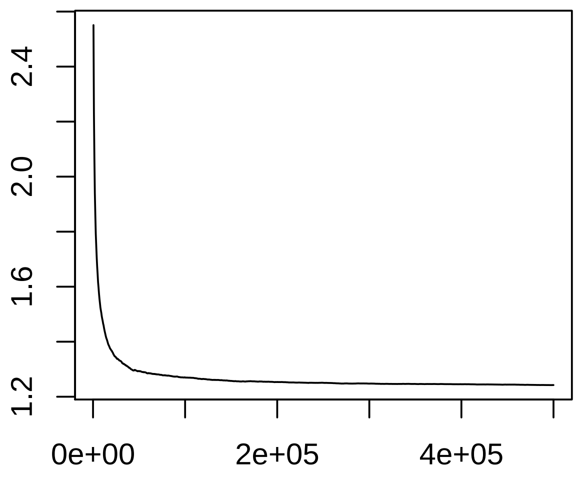

Figure 9. The log proposal standard deviation ls_2 corresponding to the SCAM variable θ_2, plotted against iteration number.

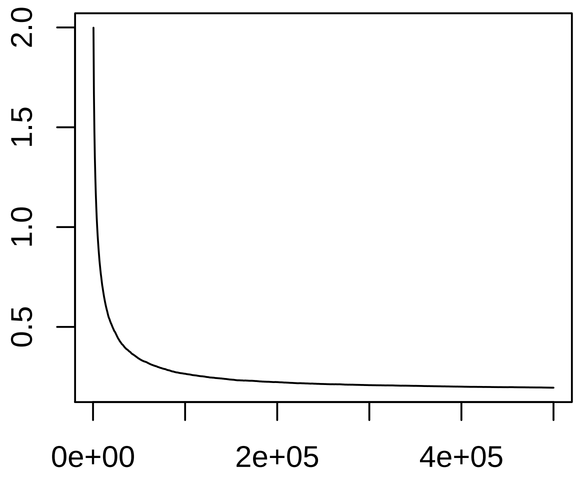

Figure 10. The log proposal standard deviation ls_3 corresponding to the SCAM variable θ_3, plotted against iteration number.

Variable	r_i	Algorithm	$\log(\sigma)$	ACT	Avr Sq Dist
θ_1	5	Adaptive	2.35	2.59	14.932
θ_1	5	Fixed	0	31.69	0.863
θ_1	5	SCAM	2.38	2.77	14.951
θ_2	50	Adaptive	1.21	2.72	1.508
θ_2	50	Fixed	0	7.33	0.581
θ_2	50	SCAM	1.27	2.77	1.486
θ_3	500	Adaptive	0.08	2.72	0.150
θ_3	500	Fixed	0	2.67	0.147
θ_3	500	SCAM	0.26	2.77	0.145

The table shows that the results of SCAM are comparable to those of our adaptive Metropolis-within-Gibbs algorithm. In this case, they were virtually identical for θ_1, and just *slightly* worse for θ_2 and θ_3. As for choice of proposal variance σ^2, there are some differences, with the SCAM choices generally larger than those for our algorithm. Overall, we feel that both of these algorithms are useful approaches to high-dimensional adaptive MCMC, and both should be kept in the applied user's arsenal.

4. State-Dependent Scaling.

We next consider examples of full-dimensional Metropolis-Hastings algorithms, where the proposal distribution is given by $Q(x,\cdot) = N(x,\,\sigma_x^2)$, i.e. such that the proposal variance depends on the current state $x \in \mathcal{X}$. For such an algorithm, according to the usual Metropolis-Hastings formula (Hastings, 1970), a proposal from x to y is accepted with probability

$$\alpha(x,y) \;=\; \min\left[1,\;\frac{\pi(y)}{\pi(x)}\,(\sigma_x/\sigma_y)^d\;\exp\big(-\frac{1}{2}(x-y)^2(\sigma_y^{-2}-\sigma_x^{-2})\big)\right]. \tag{4}$$

As a first case, we let $\mathcal{X} = \mathbf{R}$, and $\pi(\cdot) = N(0,1)$. We consider proposal kernels of the form

$$Q_{a,b}(x,\cdot) = N\left(x,\; e^a\Big(\frac{1+|x|}{\exp(\hat{\pi})}\Big)^b\right),$$

where $\hat{\pi}$ is our current empirical estimate of $\pi(g)$ where $g(x) = \log(1+|x|)$. (We divide by $\exp(\hat{\pi})$ to make the choices of a and b "orthogonal" in some sense.) After the n^{th} batch of 100 iterations, we update a by adding or subtracting $\delta(n)$ in an effort to, again, make the acceptance rate as close as possible to 0.44. We also add or subtract $\delta(n)$ to b to make the acceptance rates, acc_- and acc_+ respectively, in the regions $A_- = \{x \in \mathcal{X} : \log(1+|x|) > \hat{\pi}\}$ and $A_+ = \{x \in \mathcal{X} : \log(1+|x|) \le \hat{\pi}\}$ as equal as possible. This then increases the proposal variance in the region where acceptance rates are highest (thus lowering the acceptance rate)

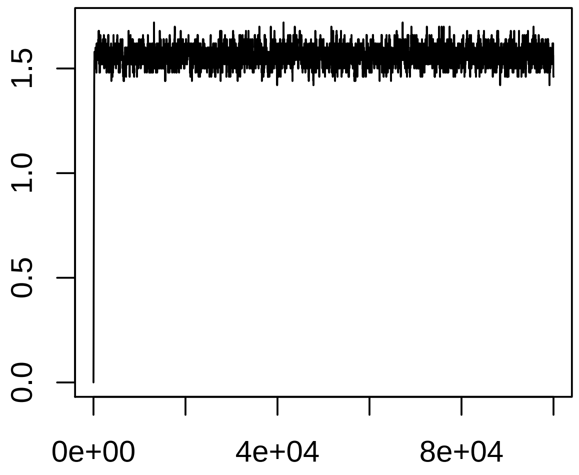

Figure 11. The tuning parameter a in the State-Dependent Scaling example, plotted against batch number, showing quick approach to "good" values near 1.5.

and correspondingly increasing the acceptance rate where the acceptance rate is lowest. The criterion being minimised here is therefore $acc_+^2 + acc_-^2$.

As in previous examples, condition (1) is automatically satisfied, at least if we insist on $\delta(n) \to 0$. In order for us to be able to demonstrated (2) however (at least for a particular family of target densities), we shall impose an extra condition, requiring that a and b be constrained within $[-M, M]$ for some global parameter $M < \infty$.

So how does this algorithm perform in practice? Empirical expected values quickly converge to their true values, showing excellent mixing. Furthermore, the tuning parameters a and b quickly find their "good" values (Figures 11 and 12), though they do continue to oscillate due to the extremely slow rate at which $\delta(n) \to 0$.

To determine how well the adaptive algorithm is performing, we compare its integrated autocorrelation time and average squared jumping distance to corresponding non-adaptive algorithms, having either fixed constant variance σ^2 (including the optimal constant value, $(2.38)^2$), and to the corresponding variable-variance algorithm. The results are as follows:

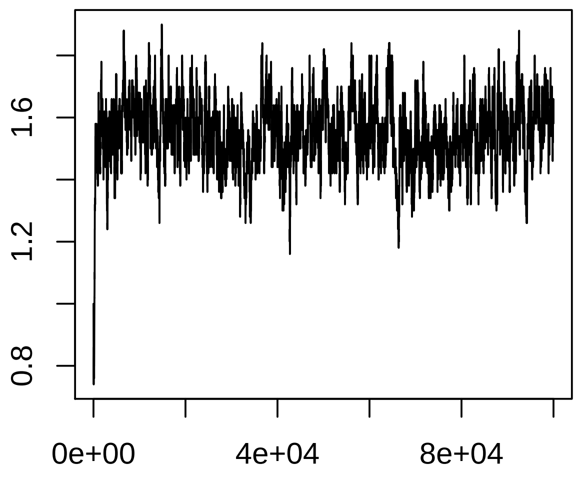

Figure 12. The tuning parameter b in the State-Dependent Scaling example, plotted against batch number, showing quick approach to "good" values near 1.6, but with significant oscillation.

Algorithm	Acceptance Rate	ACT	Avr Sq Dist
Adaptive (as above)	0.456	2.63	0.769
$\sigma^2 = \exp(-5)$	0.973	49.92	0.006
$\sigma^2 = \exp(-1)$	0.813	8.95	0.234
$\sigma^2 = 1$	0.704	4.67	0.450
$\sigma^2 = (2.38)^2$	0.445	2.68	0.748
$\sigma^2 = \exp(5)$	0.237	7.22	0.305
$\sigma_x^2 = e^{1.5}\left(\frac{1+\|x\|}{0.534822}\right)^{1.6}$	0.456	2.58	0.778

We see that our adaptive scheme is much better than arbitrarily-chosen fixed-variance algorithms, slightly better than the optimally-chosen fixed-variance algorithm (chosen by an ad-hoc search for maximising Average Square Jumping Distance, and given on the 5th line), and nearly as good as an ideally-chosen variable-σ^2 scheme chosen using a similar maximisation of average squared jumping distance on a grid of possible (a, b) values (bottom line). The results are quite impressive, since we didn't do any manual tuning of our algorithm at all other than telling the computer to seek a 0.44 acceptance rate.

While these functional forms of σ_x^2 seem promising, it is not clear how to generalise them to higher dimensional problems. Instead, we next consider a different algorithm in which the σ_x^2 are piecewise constant over various regions of the state space.

5. Regional Adaptive Metropolis Algorithm (RAMA).

The Regional Adaptive Metropolis Algorithm (RAMA) begins by partitioning the state space $\mathcal{X}$ into a finite number of disjoint regions: $\mathcal{X} = \mathcal{X}_1 \dot{\cup} \ldots \dot{\cup} \mathcal{X}_m$. The algorithm then proceeds by running a Metropolis algorithm with proposal $Q(x, \cdot) = N(x, \exp(2\, a_i))$ whenever $x \in \mathcal{X}_i$. Thus, if $x \in \mathcal{X}_i$ and $y \in \mathcal{X}_j$, then $\sigma_x^2 = e^{2a_i}$ and $\sigma_y^2 = e^{2a_j}$, and it follows from (4) that a proposal from x to y is accepted with probability

$$\alpha(x, y) \;=\; \min\left[1,\; \frac{\pi(y)}{\pi(x)}\; \exp\left(d(a_i - a_j) - \frac{1}{2}(x-y)^2[\exp(-2a_j) - \exp(-2a_i)]\right)\right].$$

The adaptions proceed as follows, in an effort to make the acceptance probability close to 0.234 in each region. (Such an acceptance rate is optimal in certain high-dimensional settings; see Roberts et al., 1997; Roberts and Rosenthal, 1998, 2001; Bédard, 2006a, 2006b, and we envisage that typically $\mathcal{X}_i$ would be a space of the same dimension as $\mathcal{X}$.) For $1 \le i \le d$, the parameter a_i is updated by, after the n^{th} batch of 100 iterations, considering the fraction of acceptances of those proposals which originated from $\mathcal{X}_i$. If that fraction is less than 0.234 then a_i is decreased by $\delta(n)$, while if it is more than a_i is increased by $\delta(n)$. Then, if $a_i > M$ we set $a_i = M$, while if $a_i < -M$ we set $a_i = -M$. Finally, if there were no proposals from $\mathcal{X}_i$ during the entire batch, then a_i is left unchanged. Thus the algorithm attempts to minimise $\sum_{i=1}^m acc_i^2$ where acc_i represents the acceptance rate for moves starting in the region $\mathcal{X}_i$.

Provided that $\delta(n) \to 0$, condition (1) will trivially hold. Moreover, if we assume that $M < \infty$ then it is natural to demonstrate (2) again by using a simultaneous drift condition. Such an argument will require some conditions on the target density, but is easy to demonstrate for log-concave densities such as the example below. See Section 8 for further discussion.

For a first example, we let $\mathcal{X} = \mathbf{R}^d$, and $\pi(\cdot) = N(0, I_d)$. We consider proposal kernels of the form

$$Q_{a,b}(x, \cdot) = N\Big(x,\; e^{2a}\, \mathbf{1}_{\|x\|^2 \le d} + e^{2b}\, \mathbf{1}_{\|x\|^2 > d}\Big).$$

Once every 100 iterations, we update a by adding or subtracting $\delta(n)$ to make the acceptance rate in the region $\{\|x\|^2 \le d\}$ as close as possible to 0.234. We also add or subtract $\delta(n)$ to b to make the acceptance rate in the region $\{\|x\| > d\}$ as close as possible to 0.234. We again restrict a and b to some $[-M, M]$. (We take $\delta(n) = \min(0.01,\, n^{-1/2}) \equiv 0.01$ and $M = 100$.) We choose dimension $d = 10$, and begin with $a = b = 0$.

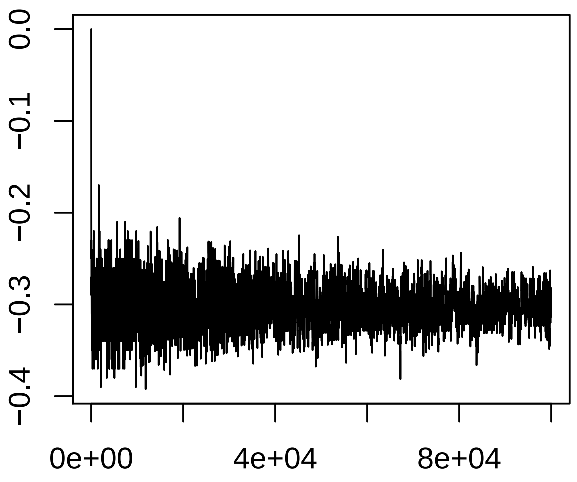

Figure 13. The tuning parameter a in the Normal RAMA example, plotted against batch number.

How well does it work? The tuning parameters a and b quickly migrate towards their "good" values of -0.3 and -0.13, respectively, but they continue to oscillate somewhat around these values (Figures 13 and 14).

How good are the values of a and b found by the computer? The following table gives comparisons of the integrated autocorrelation time and average squared jumping distance for various choices of a and b:

a, b	ACT	Avr Sq Dist
adaptive (as above)	15.54	0.1246
-0.3, -0.13	15.07	0.1258
-0.3, 0.0	15.44	0.1213
0.0, -0.13	17.04	0.1118
0.0, 0.0	17.037	0.1100

The table indicates that the adaptive algorithm (top line) is quite competitive with the corresponding fixed-parameter choice (second line), which in turn has smaller integrated autocorrelation time, and larger average squared jumping distance, than any of the other choices of a and b. This indicates that the computer has again succeeded in finding good values for the tuning parameters.

Next, we consider the following statistical model related to James-Stein estimators, as

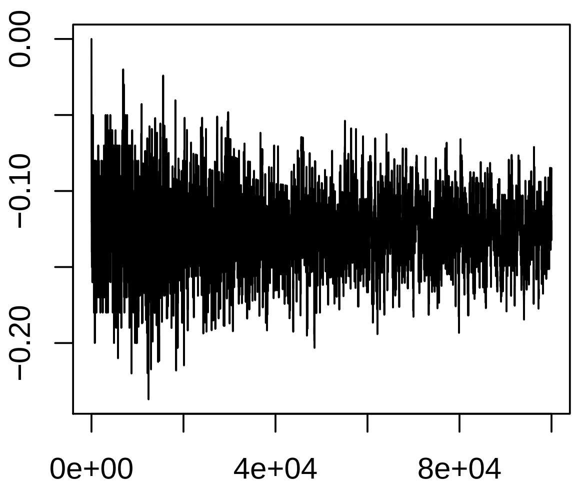

Figure 14. The tuning parameter b in the Normal RAMA example, plotted against batch number.

studied in e.g. Rosenthal (1996):

$$\begin{tikzcd}[row sep=small] & & \mu \arrow[dll] \arrow[d] \arrow[drr] & & \\ \theta_1 \arrow[d] & \cdots & \cdots & \cdots & \theta_K \arrow[d] \\ Y_1 & \cdots & \cdots & \cdots & Y_K \end{tikzcd} \qquad \begin{array}{ll} \theta_i \sim N(\mu, A) & [1 \le i \le K] \\ \\ Y_i \sim N(\theta_i, V) & [1 \le i \le K] \end{array}$$

Here the $\{Y_i\}$ are observed data. We use the prior distributions $\mu \sim N(\mu_0, \sigma_0^2)$ and $A \sim IG(a_1, b_1)$, and replace V by its (fixed) empirical Bayes estimate. We let $\pi(\cdot)$ be the resulting posterior distribution for $(A, \mu, \theta_1, \ldots, \theta_K)$, on the $(K+2)$-dimensional state space $\mathcal{X} = [0, \infty) \times \mathbf{R}^{K+1}$. The density of $\pi(\cdot)$, with respect to Lebesgue measure, is then given by

$$\begin{aligned} f(A, \mu, \theta_1, \ldots, \theta_K) \;&=\; N(\mu_0, \sigma_0^2; \mu)\; IG(a_1, b_1; A) \times \prod_{i=1}^{K} \Big[N(\mu, A; \theta_i)\; N(\theta_i, V; Y_i) \Big] \\ &\propto\; \exp(-(\mu-\mu_0)^2/2\sigma_0^2)\; \exp(-b_1/A) \,/\, A^{a_1+1} \times \\ &\qquad \times \prod_{i=1}^{K} \Big[A^{-1/2} \exp(-(\theta_i-\mu)^2/2A)\; V^{-1/2} \exp(-(Y_i-\theta_i)^2/2V) \Big]. \end{aligned}$$

For a numerical example, we let $K = 18$, and let $Y_1, \ldots, Y_{18}$ be the (real) baseball data of Table 1 of Morris (1983) (see also Efron and Morris, 1975). Thus, $\mathcal{X} \subseteq \mathbf{R}^{20}$. We choose the prior parameters as $\mu_0 = 0$, $\sigma_0^2 = 1$, $a_1 = -1$, and $b_1 = 2$.

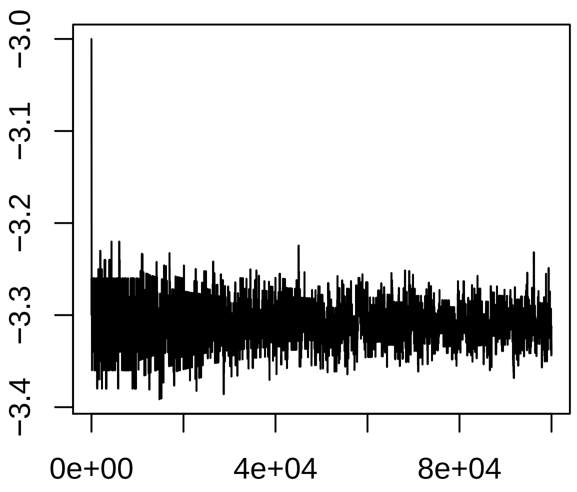

Figure 15. The tuning parameter a in the James-Stein RAMA example, plotted against batch number.

We again perform the RAMA algorithm. Specifically, after the n^{th} batch of 100 iterations, we update a by adding or subtracting $\delta(n)$ to make the acceptance rate in the region $\{\sum_i(\theta_1 - \mu_0)^2 \leq 0.15\}$ as close as possible to 0.234. We also add or subtract $\delta(n)$ to b to make the acceptance rate in the region $\{\sum_i(\theta_1 - \mu_0)^2 > 0.15\}$ as close as possible to 0.234.

The simulations again show good mixing, and rapid convergence of functional averages to their true posterior means. Furthermore, the adaptive parameters a and b quickly settle down to near -3.3 and -3.2 respectively (Figures 15, 16).

How good are the values of the tuning parameters chosen? We again compare integrated autocorrelation times and average squared jumping distances, as follows (acceptance rates are also shown):

a, b	Acc Rate	ACT	Avr Sq Dist $\times 10^4$
adaptive (as above)	0.228	31.60	2.756
-3.3, -3.2	0.194	25.75	2.793
-2.3, -2.3	0.003	50.67	0.192
-4.3, -4.3	0.655	38.92	1.168
-3.3, -4.3	0.647	36.91	1.153
-4.3, -3.3	0.281	38.04	2.407
-0.6, -0.6	2.5×10^{-5}	53.97	0.010

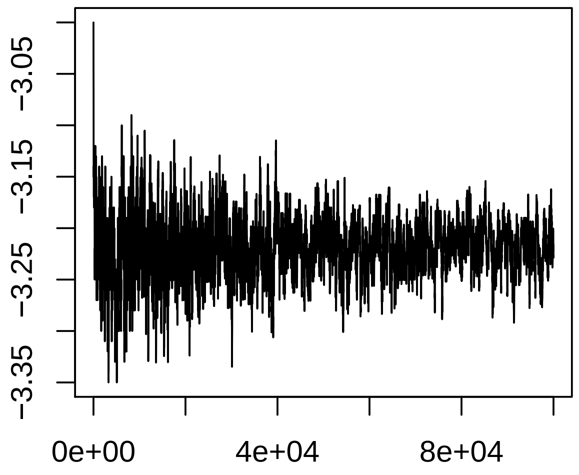

Figure 16. The tuning parameter b in the James-Stein RAMA example, plotted against batch number.

We again see that the adaptive algorithm (top line) is quite competitive with the corresponding fixed-parameter choice (second line), which in turn is better than any of the other choices of a and b. This shows that, once again, the adaptive algorithm has automatically chosen good values of the MCMC tuning parameters, without requiring user intervention.

Remarks.

1. In our simulations, the condition $M < \infty$ has never been necessary, since RAMA has never tried to push any of the $\{a_j\}$ towards unbounded values. Indeed, we conjecture that under appropriate regularity assumptions (e.g. if the densities are jointly continuous), condition (2) will be satisfied automatically due to drifting of the parameters a_i back to reasonable values due to the adaptive process (cf. Roberts and Rosenthal, 2005, Corollary 14).
2. If some value a_j is much too large, then $\alpha(x, y)$ may be very small for all $y \in \mathcal{X}_j$ and $x \notin \mathcal{X}_j$. This means that the region $\mathcal{X}_j$ may virtually never be entered, so that a_j will remain virtually constant, leading to isolation of $\mathcal{X}_j$ and thus very poor convergence. Hence, it is important with RAMA to begin with sufficiently small values of the $\{a_i\}$. Alternatively, it might be wise to decrease each a_i slightly (rather than leaving it unchanged) after each batch in which there were no proposals from $\mathcal{X}_i$.
3. The version of RAMA presented here requires that the user specify the regions $\{\mathcal{X}_i\}_{i=1}^m$

by hand. However, it may also be possible to have the computer automatically select appropriate regions, by e.g. doing a preliminary run with fixed proposal variance, and then grouping together state space subsets which appear to have similar acceptance rates.

4. Roberts et al. (1997) show that in certain situations the optimal scaling for Metropolis algorithms can be characterised as that which has acceptance probability 0.234. One can ask whether these results carry over to RAMA, and whether equal acceptance rates on different regions (as sought by RAMA) truly leads to optimality. We believe this to be true quite generally, but can only prove it for very specific settings (e.g. birth-death processes). The method of proof of Roberts et al. (see also Bédard, 2006a, 2006b) appears to carry over away from the region boundaries, but the behaviour at the region boundaries is more complicated.
5. If we set $\delta(n)$ to a constant, as opposed to having $\delta(n) \to 0$, then condition 1 might fail, so the chain might not converge to $\pi(\cdot)$. On the other hand, the chain together with the parameter values $\{a_j\}$ is jointly Markovian, and under appropriate scaling may have its own joint diffusion limit. It would be interesting (Stewart, 2006) to study that diffusion limit, to e.g. see how much asymptotic error results from failing to satisfy (1).

6. To Log or Not To Log.

Suppose π is the density function for a real-valued random variable W. Then if π is heavy-tailed, it may be advantageous to take logarithms, i.e. to instead consider the density function for $\widetilde{W} \equiv \log W$. This leads to the question, when is it advantageous to consider $\widetilde{W}$ in place of W? Once again, adaptive algorithms can provide insights into this question.

To avoid problems of negative or near-negative values, we modify the logarithm function and instead consider the function

$$\ell(w) \equiv \operatorname{sgn}(w)\ \log(1 + |w|)\,,$$

where $\operatorname{sgn}(w) = 1$ for $w > 0$, and $\operatorname{sgn}(w) = -1$ for $w < 0$. The function ℓ is an increasing, continuously differentiable mapping from $\mathbf{R}$ onto $\mathbf{R}$, with inverse $\ell^{-1}(w) = \operatorname{sgn}(w)\,(e^{|w|} - 1)$, and graph as follows:

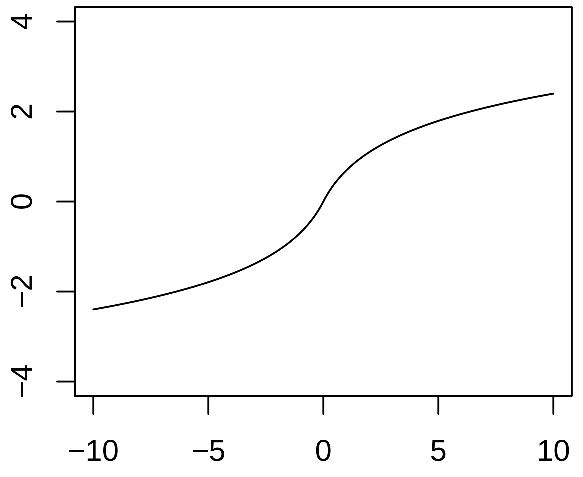

Figure 17. Graph of the modified log function ℓ.

If π is the density for W, and $\widetilde{W} = \log(W)$, then taking Jacobians shows that the density for $\widetilde{W}$ is given by $\tilde{\pi}(w) = e^{|w|}\,\pi(e^{|w|} - 1)$.

A result of Mengersen and Tweedie (1996) says, essentially, that a random-walk Metropolis (RWM) algorithm for a density π will be geometrically ergodic if and only if π has exponential or sub-exponential tails, i.e. satisfies

$$\log \pi(x) - \log \pi(y) \ \geq\ \eta(y - x)\,, \qquad y > x \geq x_1 \tag{5}$$

for some $x_1 > 0$ and $\eta > 0$ (and similarly for $y < x \leq -x_1$). (A similar result holds in multi-dimensional contexts , cf. Roberts and Tweedie, 1996.) But if π on $\mathbf{R}$ satisfies (5), then so does $\tilde{\pi}$, since if $y > x \geq -\log(\eta) + \beta \geq x_1 > 0$, then

$$\log \tilde{\pi}(x) - \log \tilde{\pi}(y) \ =\ (x - y) + \log \pi(e^x - 1) - \log \pi(e^y - 1)$$

$$\geq\ (x - y) + \eta((e^y - 1) - (e^x - 1)) \ =\ -(y - x) + \eta e^x(e^{y-x} - 1)$$

$$\geq\ -(y - x) + \eta^x(y - x) \ =\ (y - x)(\eta e^x - 1) \ \geq\ (y - x)(e^\beta - 1)\,.$$

Hence, (5) is satisfied for $\tilde{\pi}$ with $\tilde{\eta} = e^\beta - 1$. In fact, by making β arbitrarily large, we can make $\tilde{\eta}$ as large as we like, showing that the tails of $\tilde{\pi}$ are in fact sub-exponential.

This suggests that, at least as far as geometric ergodicity is concerned, it is essentially always better to work with $\tilde{\pi}$ than with π. As a specific example, if π is the standard Cauchy distribution, then RWM on π is *not* geometrically ergodic, but RWM on $\tilde{\pi}$ is.

Despite this evidence in favour of log transforms for RWM, it is not clear that taking logarithms (or applying ℓ) necessarily helps with the *quantitative* convergence of RWM. To investigate this, we use an adaptive algorithm.

Specifically, given π, we consider two different algorithms: one a RWM on π, and the other a RWM on $\tilde{\pi}$, each using proposal distributions of the form $Q(x, \cdot) = N(x, \sigma^2)$. After the n^{th} batch of 100 iterations, we allow each version to adapt its own scaling parameter σ by adding or subtracting $\delta(n)$ to $\log(\sigma)$, in an effort to achieve acceptance rate near 0.44 for each version. Then, once every 100 batches, we consider whether to switch versions (i.e., to apply ℓ if we currently haven't, or to undo ℓ if we currently have), based on whether the current average squared jumping distance is smaller than that from the last time we used the other version. (We force the switch to the other version if it fails 100 times in succession, to avoid getting stuck forever with just one version.)

How does this adaptive algorithm work in practice? In the following table we considered three different one-dimensional symmetric target distributions: a standard Normal, a standard Cauchy, and a Uniform$[-100, 100]$. For each target, we report the percentage of the time that the adaptive algorithm spent on the logged density $\tilde{\pi}$ (as opposed to the regular density π). We also report the mean value of the log proposal standard deviation for both the regular and the logged RWM versions.

Target	Log %	ls_{reg}	ls_{log}
Normal	3.62%	2.52	2.08
Cauchy	99.0%	3.49	2.66
Uniform	4.95%	6.66	2.65

We see from this table that, for the Normal and Uniform distributions, the adaptive algorithm saw no particular advantage to taking logarithms, and indeed stayed in the regular (unlogged) π version the vast majority of the time. On the other hand, for the Cauchy target, the algorithm uses the logged $\tilde{\pi}$ essentially as much as possible. This shows that this adaptive algorithm is able to distinguish between when taking logs is helpful (e.g. for the heavy-tailed Cauchy target), and when it is not (e.g. for the light-tailed Normal and Uniform targets).

For multi-dimensional target distributions, it is possible to take logs (or apply the function ℓ) separately to each coordinate. Since lighter tails are still advantageous in multi-dimensional settings (Roberts and Tweedie, 1996), it seems likely to be advantageous to apply ℓ to precisely those coordinates which correspond to heavy tails in the target distribution. In high dimensions, this cannot feasibly be done by hand, but an adaptive algorithm could still do it automatically. Such multi-dimensional versions of this adaptive logarithm algorithm appear worthy of further investigation.

7. How to adapt.

Whilst the theory of adaptation has progressed significantly in recent years, practical implementation raises many important and largely unstudied problems. One issue is that we still know very little about how to optimise MCMC algorithms. The use of monitored acceptance probabilities has the appeal of simplicity and the support of some theory. In 1-dimension the use of 0.44 and in higher-dimensional problems the adoption of 0.234 together with the scaling rule $\sigma = 2.38/d^{1/2}$ are based on results in Gelman et al (1996). However the use of these simple rules, although often effective, are based on approximations and are not rigorously proved for complex non-homogeneous models used in statistical analysis. In our two heterogeneous scaling examples, we are guided by established theoretical properties of MCMC - particularly that Metropolis algorithms are not geometrically ergodic for heavy tailed target densities. We believe that there is considerable further scope for algorithm development based on MCMC theory.

One important question asks whether an effective adaptive scheme should require that Γ converges. It is intuitively appealing to think of the adaptive scheme searching for "the best" algorithm from a collection of candidates. Our approach here is, however, not to require convergence of Γ since we are eager to have adaptive procedures which work in as general a context as possible. It may well be (and we suspect so) that all the examples in this paper involve situations where Γ_n does converge, but we have not attempted to demonstrate this. A complementary approach to our work in this respect is that adopted by Andrieu and Moulines (2003). This has the appeal of generality, but it may be that an algorithm in which we do not have convergence of Γ_n converges less rapidly than one in which Γ_n does converge. More experience with practical examples is necessary to resolve these issues.

AM and RAMA are set up naturally in a multi-dimensional context. Multivariate generalisations of the state-dependent strategy used in Section 4 are clearly possible. The simplest idea is to apply the same strategy to each of the d components, obtaining a collection of parameters, $(a_1, b_1) \ldots (a_d, b_d)$ defining a proposal of independent components with variance in the ith direction given by $e^a(1 + |x_i|)^b$. More complex proposals which try to respect the dependence in the target density (as in AM) are also possible.

8. Checking the Bounded Convergence condition.

The Diminishing Adaptation condition is relatively easy to check, and in fact adaptive procedures are generally constructed with this condition directly in mind. On the other hand, the Bounded Convergence condition is typically more difficult to check.

One way of showing this is to show that all MCMC kernels satisfy the same Lyapunov drift condition. For instance, Roberts and Rosenthal (2005) show that an adaptive MCMC algorithm satisfying Diminishing Adaptation satisfies Bounded Convergence (and is hence ergodic) if the family $\{P_\gamma\}_{\gamma\in\mathcal{Y}}$ *simultaneously strongly aperiodically geometrically ergodic*, ie there is $C \in \mathcal{F}$, $V : \mathcal{X} \to [1,\infty)$, $\delta > 0$, $\lambda < 1$, and $b < \infty$, such that $\sup_C V = v < \infty$, and
(i) for each $\gamma \in \mathcal{Y}$, there exists a probability measure $\nu_\gamma(\cdot)$ on C with $P_\gamma(x,\cdot) \geq \delta\,\nu_\gamma(\cdot)$ for all $x \in C$; and
(ii) $(P_\gamma)V \leq \lambda V + b\,\mathbf{1}_C$.

A natural approach to the establishment of simultaneously strongly aperiodically geometrically ergodicity is to use the drift function $\pi^{-1/2}$ as in Roberts and Tweedie (1996) for AM and Roberts and Rosenthal (1997) for Adaptive Metropolis-within-Gibbs. As an example of a precise result which can be shown in this way, for the AM algorithm, the condition will hold by this argument for all target densities which are log-concave (except perhaps on some bounded region).

The state-dependent proposal variance case can also be analysed to give a drift condition with Lyapunov function $\pi^{-1/2}$, at least for $b < 2$. This is essentially because asymptotically (as $|x| \to \infty$) the accept/reject ratio is dominated by the ratio $\pi(y)/\pi(x)$ and thus all moves to smaller $|x|$ values are accepted with all moves to larger $\pi(x)$ values are possibly rejected. Then standard calculation as those in Roberts and Tweedie (1996), together with continuity and compactness arguments for the parameters a and b are sufficient to demonstrate the simultaneous drift condition.

It seems that these results will be easily generalisable to the other examples in this paper, essentially because all methods are essentially constructed from random walk Metropolis. These extensions are subject to further work (Bai, Roberts, and Rosenthal, 2008).

9. Conclusion.

This paper has considered automated tuning of MCMC algorithms, especially Metropolis-Hastings algorithms, with quite positive results.

For example, for Metropolis-within-Gibbs algorithms, the following (generally well-known) statements are all reinforced through our simulation. (1) The choice of proposal variance

σ^2 is crucial to the success of the algorithm. (2) Good values of σ^2 can vary greatly from one coordinate to the next. (3) There are far too many coordinates to be able to select good values of σ^2 for each coordinate by hand. (4) Adaptive methods can be used to get the computer to find good values of σ^2 automatically. (5) If done carefully, the adaptive methods can be provably ergodic, and quite effective in practice, thus allowing for good tuning and rapid convergence of MCMC algorithms that would otherwise be impractical.

The practical experience of this paper, though very promising, also raises important questions. In particular, how robust are the strategies suggested in the various methods here? For example, in the adaptive Metropolis-within-Gibbs example, how crucial is the choice of $\delta(n)$ in the success of the method, and how does this vary from problem to problem?

We still know comparatively little about what adaptive strategies to use in any particular context. Our feeling is that the choice of adaptive strategy should be guided by theoretical knowledge about MCMC. For instance, when using RWM, particular problems are observed with heavy-tailed target distributions (such as lack of geometric ergodicity, breakdown of CLTs etc). In this case it makes sense to use a strategy which attempts to stabilise the algorithm excursions, and this points to the use of heterogeneous scaling and/or a strategy which lightens the tails (such as that introduced in Section 6).

One important issue for adaptive scaling concerns the practical issue that scaling the proposal correlation structure to match that of the target will be a very poor strategy when some of the target distribution variances are infinite. Typically in practical MCMC situations, this may not be very easy to check analytically by inspection of the target density. For this reason, perhaps it makes more sense to scale according to acceptance rate criteria rather than variances. However it is impossible to use this to match correlation structure, and further work is required to introduce robust versions of the AM and other methods.

Adaptive strategies are generally simple to implement. However it is very important that such a strategy is constructed in such a way that the conditions for ergodicity are satisfied. Further work is clearly required to give sufficiently simple conditions to enable routine adaptation to take place in applied problems. In terms of adaptive strategies, there is now extensive MCMC theory to help guide the construction of suitable adaptive algorithms. One potential problem evolves from an adaptive strategy which is too "greedy" in that it tries to adapt too closely to initial information from the output. Such algorithms can take considerable time to "recover" from misleading initial information.

Overall, we feel that these results indicate the widespread applicability of adaptive MCMC algorithms to many different MCMC settings, including complicated high-dimensional distributions. We hope that this paper will inspire users of MCMC to experiment with adaptive algorithms in their future applications (e.g. Turro et al., 2007). All of the software used to run the algorithms described herein is freely available at probability.ca/adapt.

Acknowledgements. We thank Sylvia Richardson for a very helpful suggestion, and thank the editors and referees for many constructive comments that greatly improved the paper.

REFERENCES

C. Andrieu and Y.F. Atchadé (2005), On the efficiency of adaptive MCMC algorithms. Preprint.

C. Andrieu and E. Moulines (2003), On the ergodicity properties of some adaptive Markov Chain Monte Carlo algorithms. Preprint.

Y.F. Atchadé and J.S. Rosenthal (2005), On Adaptive Markov Chain Monte Carlo Algorithms. Bernoulli **11**, 815–828.

Y. Bai, G.O. Roberts, and J.S. Rosenthal (2008), On the containment condition for adaptive Markov chain Monte Carlo algorithms. Preprint.

M. Bédard (2006a), Weak convergence of Metropolis algorithms for non-iid target distributions. Preprint.

M. Bédard (2006b), Optimal acceptance rates for Metropolis algorithms: moving beyond 0.234. Preprint.

A.E. Brockwell and J.B. Kadane (2005), Identification of regeneration times in MCMC simulation, with application to adaptive schemes. J. Comp. Graph. Stat. **14**, 436–458.

B. Efron and C. Morris (1975), Data analysis using Stein's estimator and its generalizations. J. Amer. Stat. Assoc., Vol. **70**, No. **350**, 311-319.

A.E. Gelfand and A.F.M. Smith (1990), Sampling based approaches to calculating marginal densities. J. Amer. Stat. Assoc. **85**, 398-409.

W.R. Gilks, G.O. Roberts, and S.K. Sahu (1998), Adaptive Markov Chain Monte Carlo. J. Amer. Stat. Assoc. **93**, 1045–1054.

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

H. Haario, E. Saksman, and J. Tamminen (2001), An adaptive Metropolis algorithm. Bernoulli **7**, 223–242.

H. Haario, E. Saksman, and J. Tamminen (2005), Componentwise adaptation for high dimensional MCMC. Comput. Stat. **20**, 265–274.

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

K.L. Mengersen and R.L. Tweedie (1996), Rates of convergence of the Hastings and Metropolis algorithms. Ann. Statist. **24**, 101–121.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller (1953), Equations of state calculations by fast computing machines. J. Chem. Phys. **21**, 1087–1091.

C. Morris (1983), Parametric empirical Bayes confidence intervals. Scientific Inference, Data Analysis, and Robustness, 25-50.

G.O. Roberts, A. Gelman, and W.R. Gilks (1997), Weak convergence and optimal scaling of random walk Metropolis algorithms. Ann. Appl. Prob. **7**, 110–120.

G.O. Roberts and J.S. Rosenthal (1998), Optimal scaling of discrete approximations to Langevin diffusions. J. Roy. Stat. Soc. B **60**, 255–268.

G.O. Roberts and J.S. Rosenthal (2001), Optimal scaling for various Metropolis-Hastings algorithms. Stat. Sci. **16**, 351–367.

G.O. Roberts and J.S. Rosenthal (2005), Coupling and Ergodicity of Adaptive MCMC. Preprint.

G.O. Roberts and R.L. Tweedie (1996), Geometric Convergence and Central Limit Theorems for Multidimensional Hastings and Metropolis Algorithms. Biometrika **83**, 95–110.

J.S. Rosenthal (1996), Analysis of the Gibbs sampler for a model related to James-Stein estimators. Stat. and Comp. **6**, 269–275.

J.S. Rosenthal (2004), Adaptive MCMC Java Applet. Available at:

http://probability.ca/jeff/java/adapt.html

A. Stewart (2006), Personal communication.

L. Tierney (1994), Markov chains for exploring posterior distributions (with discussion). Ann. Stat. **22**, 1701–1762.

E. Turro, N. Bochkina, A.M.K. Hein, and S. Richardson (2007), BGX: a Bioconductor package for the Bayesian integrated analysis of Affymetrix GeneChips. BMC Bioinformatics **8**, 439–448. Available at: http://www.biomedcentral.com/1471-2105/8/439