Optimal Proposal Distributions and Adaptive MCMC

by

Jeffrey S. Rosenthal*

[Chapter for MCMC Handbook, S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, eds.]

(June 2008; revised January 2010.)

Abstract. We review recent work concerning optimal proposal scalings for Metropolis-Hastings MCMC algorithms, and adaptive MCMC algorithms for trying to improve the algorithm on the fly.

1. Introduction.

The Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970) requires choice of proposal distributions, and it is well-known that some proposals work much better than others. Determining which proposal is best for a particular target distribution is both very important and very difficult. Often this problem is attacked in an *ad hoc* manner involving much trial-and-error. However, it is also possible to use theory to estimate optimal proposal scalings and/or adaptive algorithms to attempt to find good proposals automatically. This chapter reviews both of these possibilities.

1.1. The Metropolis-Hastings Algorithm.

Suppose our target distribution has density π with respect to some reference measure (usually d-dimensional Lebesgue measure). Then given $\mathbf{X}_n$, a "proposed value" $\mathbf{Y}_{n+1}$ is generated from some pre-specified density $q(\mathbf{X}_n, \mathbf{y})$, and is then accepted with probability

$$\alpha(\mathbf{x},\mathbf{y}) = \begin{cases} \min\{\frac{\pi(\mathbf{y})}{\pi(\mathbf{x})}\frac{q(\mathbf{y},\mathbf{x})}{q(\mathbf{x},\mathbf{y})}, 1\}, & \pi(\mathbf{x})\, q(\mathbf{x},\mathbf{y}) > 0 \\ 1, & \pi(\mathbf{x})\, q(\mathbf{x},\mathbf{y}) = 0\ . \end{cases} \tag{1}$$

*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.

If the proposed value is accepted, we set, $\mathbf{X}_{n+1} = \mathbf{Y}_{n+1}$; otherwise, we set $\mathbf{X}_{n+1} = \mathbf{X}_n$. The function $\alpha(\mathbf{x}, \mathbf{y})$ is chosen, of course, precisely to ensure that the Markov chain $\mathbf{X}_0, \mathbf{X}_1, \ldots$ is reversible with respect to the target density $\pi(\mathbf{y})$, so that the target density is stationary for the chain. If the proposal is *symmetric*, i.e. $q(\mathbf{x}, \mathbf{y}) = q(\mathbf{y}, \mathbf{x})$, then this reduces to

$$\alpha(\mathbf{x}, \mathbf{y}) = \begin{cases} \min\{\frac{\pi(\mathbf{y})}{\pi(\mathbf{x})}, 1\}, & \pi(\mathbf{x})\, q(\mathbf{x}, \mathbf{y}) > 0 \\ 1, & \pi(\mathbf{x})\, q(\mathbf{x}, \mathbf{y}) = 0 \;. \end{cases}$$

1.2. Optimal Scaling.

It has long been recognised that the choice of the proposal density $q(\mathbf{x}, \mathbf{y})$ is crucial to the success (e.g., rapid convergence) of the Metropolis-Hastings algorithm. Of course, the fastest-converging proposal density would be $q(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{y})$ (in which case $\alpha(\mathbf{x}, \mathbf{y}) \equiv 1$, and the convergence is immediate), but in the MCMC context we assume that π cannot be sampled directly. Instead, the most common case (which we focus on here) involves a *symmetric random-walk Metropolis algorithm* (RMW) in which the proposal value is given by $\mathbf{Y}_{n+1} = \mathbf{X}_n + \mathbf{Z}_{n+1}$, where the increments $\{\mathbf{Z}_n\}$ are i.i.d. from some fixed symmetric distribution (e.g., $N(0, \sigma^2 I_d)$). In this case, the crucial issue becomes how to *scale* the proposal (e.g., how to choose σ): too small and the chain will move too slowly; too large and the proposals will usually be rejected. Instead, we must avoid both extremes (we sometimes refer to this as the "Goldilocks Principle").

Metropolis et al. (1953) recognised this issue early on, when they considered the case $\mathbf{Z}_n \sim \text{Uniform}[-\alpha, \alpha]$ and noted that "the maximum displacement α must be chosen with some care; if too large, most moves will be forbidden, and if too small, the configuration will not change enough. In either case it will then take longer to come to equilibrium."

In recent years, significant progress has been made in identifying optimal proposal scalings, in terms of such tangible values as asymptotic acceptance rate. Under certain conditions, these results can describe the optimal scaling precisely. These issues are discussed in Section 2 below.

1.3. Adaptive MCMC.

The search for improved proposal distributions is often done manually, through trial and error, though this can be difficult especially in high dimensions. An alternative approach is adaptive MCMC, which asks the computer to automatically "learn" better parameter values "on the fly", i.e. while an algorithm runs. (Intuitively, this approach is attractive since computers are getting faster and faster, while human speed is remaining about the same.)

Suppose $\{P_\gamma\}_{\gamma \in \mathcal{Y}}$ is a family of Markov chains, each having stationary distribution π. (For example, perhaps P_γ corresponds to a RWM algorithm with increment distribution $N(0, \gamma^2 I_d)$.) An adaptive MCMC algorithm would randomly update the value of γ at each iteration, in an attempt to find the best value. Adaptive MCMC has been applied in a variety of contexts (e.g. Haario et al., 2001; Roberts and Rosenthal, 2006; Giordani and Kohn, 2006) including to problems in statistical genetics (Turro et al., 2007).

Counterintuitively, adaptive MCMC algorithms may not always preserve the stationarity of π. However, if the adaptions are designed to satisfy certain conditions, then stationarity is guaranteed, and significant speed-ups are possible. These issues are discussed in Section 3 below.

1.4. Comparing Markov Chains.

Since much of what follows will attempt to find "better" or "best" MCMC samplers, we pause to consider what it means for one Markov chain to be better than another.

Suppose P_1 and P_2 are two Markov chains, each with the same stationary distribution π. Then P_1 *converges faster than* P_2 if $\sup_A |P_1^n(x,A) - \pi(A)| \leq \sup_A |P_2^n(x,A) - \pi(A)|$ for all n and x. This definition concerns distributional convergence (in total variation distance) as studied theoretically in e.g. Rosenthal (1995, 2002) and Roberts and Tweedie (1999).

Alternatively, P_1 *has smaller variance than* P_2 if $\mathrm{Var}(\frac{1}{n}\sum_{i=1}^n g(X_i))$ is smaller when $\{X_i\}$ follows P_1 than when it follows P_2. This definition concerns variance of a functional g, and may depend on which g is chosen, and also perhaps on n and/or the starting distribution. Usually we assume the Markov chain $\{X_n\}$ is in stationarity, so $\mathbf{P}(X_i \in A) = \pi(A)$, and $\mathbf{P}(X_{i+1} \in A \,|\, X_i = x) = P(x, A)$ where P is the Markov chain kernel being followed.

If the Markov chain $\{X_n\}$ is in stationarity, then for large n, $\mathrm{Var}(\frac{1}{n}\sum_{i=1}^n g(X_i)) \approx \frac{1}{n}\mathrm{Var}_\pi(g)\,\tau_g$, where $\tau_g = \sum_{k=-\infty}^{\infty} \mathrm{Corr}(g(X_0), g(X_k)) = 1 + 2\sum_{i=1}^{\infty} \mathrm{Corr}(g(X_0), g(X_i))$ is the integrated autocorrelation time. So, a related definition is that P_1 *has smaller asymptotic variance than* P_2 if τ_g is smaller under P_1 than under P_2. (Under strong conditions involving the so-called *Peskun ordering*, this improvement is sometimes uniform over choice of g; see e.g. Mira, 2001.)

Another perspective is that a Markov chain is better if it allows for faster exploration of the state space. Thus, P_1 *mixes faster than* P_2 if $\mathbf{E}[(X_n - X_{n-1})^2]$ is larger under P_1 than under P_2, where again $\{X_n\}$ is in stationarity. (Of course, $\mathbf{E}[(X_n - X_{n-1})^2]$ would usually be estimated by $\frac{1}{n}\sum_{i=1}^n (X_i - X_{i-1})^2$, or perhaps by $\frac{1}{n-B}\sum_{i=B}^n (X_i - X_{i-1})^2$ to allow a burn-in B to approximately converge to stationarity.) Note that the evaluation of $\mathbf{E}[(X_n - X_{n-1})^2]$ is over all proposed moves, including rejected ones where $(X_n - X_{n-1})^2 = 0$. Thus, rejected moves slow down the chain, but small accepted moves don't help too much either. Best is to find reasonably large proposed moves which are reasonably likely to be accepted.

Such competing definitions of "better" Markov chain mean that the optimal choice of MCMC may depend on the specific question being asked. However, we will see in Section 2 that in some circumstances, these different definitions are all equivalent, leading to uniformly optimal choices of algorithm parameters.

2. Optimal Scaling of Random-Walk Metropolis.

We restrict ourselves to the RWM algorithm, where the proposals are of the form $\mathbf{Y}_{n+1} = \mathbf{X}_n + \mathbf{Z}_{n+1}$, where $\{\mathbf{Z}_i\}$ are i.i.d. with fixed symmetric density, with some scaling parameter $\sigma > 0$, e.g. $Z_i \sim N(0, \sigma^2 I_d)$. To avoid technicalities, we assume that the target density π is a positive, continuous function. The task is to choose σ in order to optimise the resulting MCMC algorithm.

2.1. Basic Principles.

A first observation is that if σ is very small, then virtually all proposed moves will be accepted, but they will represent very small movements, so overall the chain will not mix well (Figure 1).

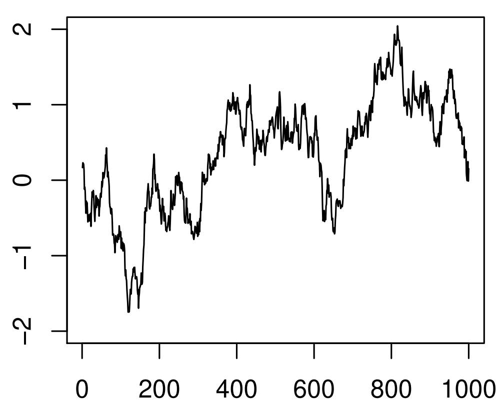

Figure 1. Trace plot with small σ, large acceptance rate, poor mixing.

Similarly, if σ is very large, then most moves will be rejected, so the chain will usually not move at all (Figure 2).

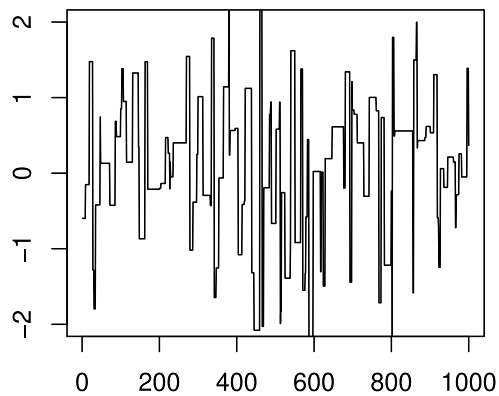

Figure 2. Trace plot with large σ, small acceptance rate, poor mixing.

What is needed is a value of σ between the two extremes, thus allowing for reasonable-sized proposal moves together with a reasonably high acceptance probability (Figure 3).

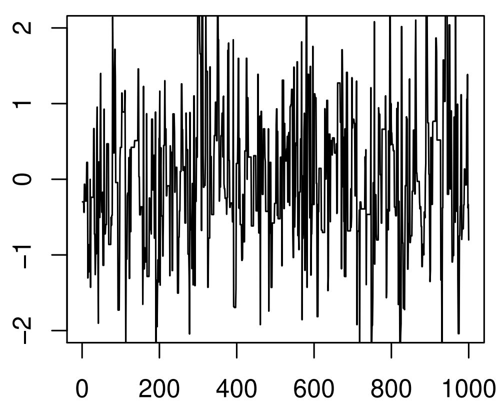

Figure 3. Trace plot with medium σ, medium acceptance rate, good mixing.

A simple way to avoid the extremes is to monitor the *acceptance rate* of the algorithm, i.e. the fraction of proposed moves which are accepted. If this fraction is very close to 1, this suggests that σ is too small (as in Figure 1). If this fraction is very close to 0, this suggests that σ is too large (as in Figure 2). But if this fraction is far from 0 and far from 1, then we have managed to avoid both extremes (Figure 3).

So, this provides an easy rule-of-thumb for scaling random-walk Metropolis algorithms: choose a scaling σ so that the acceptance rate is far from 0 and far from 1. However, this still allows for a wide variety of choices. Under some conditions, much more can be said.

2.2. Optimal Acceptance Rate as $d \to \infty$.

Major progress about optimal scalings was made by Roberts, Gelman, and Gilks (1997). They considered RWM on $\mathbf{R}^d$ for very special target densities, of the form

$$\pi(x_1, x_2, \ldots, x_d) \; = \; f(x_1)\, f(x_2) \ldots f(x_d) \tag{2}$$

for some one-dimensional smooth density f. That is, the target density is assumed to consist of i.i.d. components. Of course, this assumption is entirely unrealistic for MCMC, since it means that to sample from π it suffices to sample each component separately from the one-dimensional density f (which is generally easy to do numerically).

Under this restrictive assumption, and assuming proposal increment distributions of the form $N(0, \sigma^2 I_d)$, Roberts et al. (1997) proved the remarkable result that as $d \to \infty$, *the optimal acceptance rate is precisely 0.234*. This is clearly a major refinement of the general principle that the acceptance rate should be far from 0 and far from 1.

More precisely, their result is the following. Suppose $\sigma = \ell/\sqrt{d}$ for some $\ell > 0$. Then as $d \to \infty$, if time is speeded up by a factor of d, and space is shrunk by a factor of $\sqrt{d}$, then each component of the Markov chain converges to a diffusion having stationary distribution f, and speed function given by $h(\ell) \; = \; 2\,\ell^2\,\Phi\left(-\frac{\sqrt{I}\ell}{2}\right)$. (Here Φ is the cdf of a standard normal, and I is a constant depending on f, in fact $I = \int_{-\infty}^{\infty} \left[\left(\frac{f'(X)}{f(X)}\right)^2\right] f(x)\, dx$.)

It follows that this diffusion is optimised (in terms of *any* of the criteria of Subsection 1.4) when ℓ is chosen to maximise $h(\ell)$. It is computed numerically that this optimal value of ℓ is given by $\ell_{opt} \doteq 2.38/\sqrt{I}$.

Furthermore, the asymptotic (stationary) acceptance rate is given by $A(\ell) = 2\,\Phi\left(-\sqrt{I}\ell/2\right)$. Hence, the optimal acceptance rate is equal to $A(\ell_{opt}) \doteq 2\,\Phi(-2.38/2) \doteq 0.234$, which is where the figure 0.234 comes from.

Figure 4 plots $h(\ell)$ versus ℓ, and Figure 5 plots $h(\ell)$ versus $A(\ell)$. (We take $I = 1$ for definiteness, but any other value of I would simply multiple all the values by a constant.) In particular, the relative speed $h(\ell)$ remains fairly close to its maximum as long as ℓ is within, say, a factor of 2 of its optimal value. Equivalently, the algorithm remains relatively efficient as long as the asymptotic acceptance rate $A(\ell)$ is between, say, 0.1 and 0.6.

Of course, the above results are all asymptotic as $d \to \infty$. Numerical studies (e.g. Gelman et al., 1996; Roberts and Rosenthal, 2001) indicate that the limiting results do seem to well approximate finite-dimensional situations for d as small as five. On the other hand, they do not apply to e.g. one-dimensional increments; numerical studies on normal distributions show that when $d = 1$, the optimal acceptance rate is approximately 0.44. Finally, these results are all on continuous spaces, but there have also been studies of optimal scaling for discrete Metropolis algorithms (Neal et al., 2007).

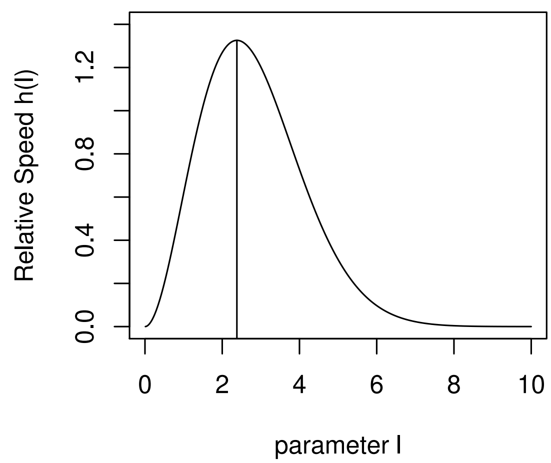

Figure 4. Algorithm relative speed $h(\ell)$ as a function of the parameter ℓ.

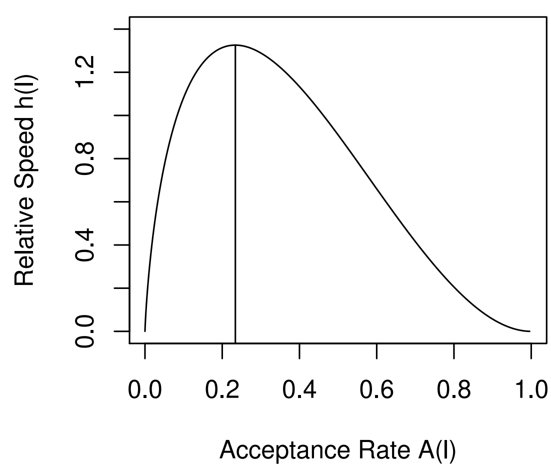

Figure 5. Algorithm relative speed $h(\ell)$ as a function of acceptance rate $A(\ell)$.

2.3. Inhomogeneous Target Distributions.

The above result of Roberts et al. (1997) requires the strong assumption that $\pi(\mathbf{x}) = \prod_{i=1}^d f(x_i)$, i.e. that the target distribution has i.i.d. components. In later work, this assumption was relaxed in various ways.

Roberts and Rosenthal (2001) considered inhomogeneous target densities of the form

$$\pi(\mathbf{x}) = \prod_{i=1}^{d} C_i\, f(C_i x_i)\,, \tag{3}$$

where $\{C_i\}$ are themselves i.i.d. from some fixed distribution. (Thus, (2) corresponds to the special case where the C_i are constant.) They proved that in this case, the result of Roberts et al. (1997) still holds (including the optimal acceptance rate of 0.234), except that the limiting diffusion speed is divided by an "inhomogeneity factor" of $b \equiv \mathbf{E}(C_i^2)/(\mathbf{E}(C_i))^2 \geq 1$. In particular, the more inhomogeneous the target distribution, i.e. the more variability of the C_i, the slower the resulting algorithm.

As a special case, if the target distribution is $N(0, \Sigma)$ for some d-dimensional covariance matrix Σ, and the increment distribution is of the form $N(0, \Sigma_p)$, then by change-of-basis this is equivalent to the case of proposal increment $N(0, I_d)$ and target distribution $N(0, \Sigma\, \Sigma_p^{-1})$. In the corresponding eigenbasis, this target distribution is of the form (3) where now $C_i = \sqrt{\lambda_i}$, with $\{\lambda_i\}_{i=1}^d$ the eigenvalues of the matrix $\Sigma\, \Sigma_p^{-1}$. For large d, this approximately corresponds to the case where the $\{C_i\}$ are random with $\mathbf{E}(C_i) = \frac{1}{d}\sum_{j=1}^d \sqrt{\lambda_j}$ and $\mathbf{E}(C_i^2) = \frac{1}{d}\sum_{j=1}^d \lambda_j$. The inhomogeneity factor b then becomes

$$b \;\equiv\; \mathbf{E}(C_i^2)/(\mathbf{E}(C_i))^2 \;\approx\; \frac{\frac{1}{d}\sum_{j=1}^d \lambda_j}{\left(\frac{1}{d}\sum_{j=1}^d \sqrt{\lambda_j}\right)^2} \;=\; d\,\frac{\sum_{j=1}^d \lambda_j}{\left(\sum_{j=1}^d \sqrt{\lambda_j}\right)^2}\,, \tag{4}$$

with $\{\lambda_j\}$ the eigenvalues of $\Sigma\, \Sigma_p^{-1}$. This expression is maximised when the $\{\lambda_j\}$ are constant, i.e. when $\Sigma\, \Sigma_p^{-1}$ is a multiple of the identity, i.e. when Σ_p is proportional to Σ.

We conclude: with increment distribution $N(0, \Sigma_p)$, and target distribution $N(0, \Sigma)$, it is best if Σ_p is approximately proportional to Σ, i.e. $\Sigma_p \approx k\, \Sigma$ for some $k > 0$. If not, this will lead to additional slow-down by the factor b.

Once we fix $\Sigma_p = k\, \Sigma$, then we can apply the original result of Roberts et al., to conclude that the optimal constant k is then $(2.38)^2/d$. That is, it is optimal to have

$$\Sigma_p \;=\; [(2.38)^2/d]\, \Sigma\,. \tag{5}$$

In a related direction, Bédard (2007, 2008, 2006; see also Bédard and Rosenthal, 2008) considered the case where the target distribution π has independent coordinates with vastly different scalings (i.e., different powers of d as $d \to \infty$). She proved that if each individual component is dominated by the sum of all components, then the optimal acceptance rate of 0.234 still holds. In cases where one component is comparable to the sum, the optimal acceptance rate is in general *less* (not more!) than 0.234. Sherlock (2006) did explicit finite-dimensional computations for the case of normal target distributions, and came to similar conclusions.

2.4. Metropolis-Adjusted Langevin Algorithm (MALA).

Finally, Roberts and Tweedie (1996) and Roberts and Rosenthal (1998) considered the more sophisticated *Metropolis-Adjusted Langevin algorithm.* This algorithm is similar to RWM, except that the proposal increment distribution $Z_i \sim N(0, \sigma^2 I_d)$ is replaced by

$$Z_i \;\sim\; N(\frac{\sigma^2}{2}\nabla \log \pi(X_n),\ \sigma^2 I_d)\,.$$

Here the extra term $\frac{\sigma^2}{2}\nabla \log \pi(X_n)$, corresponding to the discrete-time approximation to the continuous-time Langevin diffusion for π, is an attempt to move in the direction in which the (smooth) target density π is increasing.

Roberts and Rosenthal (1998) proved that in this case, under the same i.i.d. target assumption (2), a similar optimal scaling result holds. This time the scaling is $\sigma = \ell/d^{1/6}$ (as opposed to $\ell/\sqrt{d}$), and the optimal value ℓ_{opt} has the optimal asymptotic acceptance rate $A(\ell_{opt}) = 0.574$ (as opposed to 0.234).

This proves that the optimal proposal scaling σ and the acceptance rate are both significantly larger for MALA than for RWM, indicating that MALA an improved algorithm with faster convergence. The catch, of course, is that the gradient of π must be computed at each new state reached, which could be difficult and/or time-consuming. Thus, RWM is much more popular than MALA in practice.

2.5. Numerical Examples.

Here we consider some simple numerical examples, in dimension $d = 10$. In each case, the target density π is that of a ten-dimensional normal with some covariance matrix Σ, and we consider various forms of random-walk Metropolis (RMW) algorithms.

2.5.1. Off-Diagonal Covariance.

Let M be the $d \times d$ matrix having diagonal elements 1, and off-diagonal elements given by the product of the row and column number divided by d^2, i.e. $m_{ii} = 1$, and $m_{ij} = ij/d^2$

for $j \neq i$. Then let $\Sigma^{-1} = M^2$ (since M is symmetric, Σ is positive-definite), and let the target density π be that of $N(0, \Sigma)$. (Equivalently, π is such that $\mathbf{X} \sim \pi$ if $\mathbf{X} = M\,Z$ where Z is a 10-tuple of i.i.d. univariate standard normals.)

We compute numerically that the top-left entry of Σ is equal to 1.0305. So, if h is the functional equal to the square of the first coordinate, then in stationarity the mean value of h should be 1.0305.

We consider a random-walk Metropolis (RWM) algorithm for this target $\pi(\cdot)$, with initial value $X_0 = (1, 0, 0, \ldots, 0)$, and with increment distribution given by $N(0, \sigma^2 I_d)$ for various choices of σ. For each choice of σ, we run the algorithm for 100,000 iterations, and average all the values of the square of the first coordinate to estimate its stationary mean. We repeat this 10 times for each σ, to compute a sample standard error (over the 10 independent runs) and a root mean squared error (RMSE) for each choice of σ. Our results are as follows:

σ	mean acc. rate	estimate	RMSE
0.1	0.836	0.992 ± 0.066	0.074
0.7	0.230	1.032 ± 0.019	0.018
3.0	0.002	1.000 ± 0.083	0.085

We see from this table that the value $\sigma = 0.1$ is too small, leading to an overly high acceptance rate (83.6%), a poor estimate (0.992) of the mean functional value with large standard error (0.066) and large root mean squared error (0.074). Similarly, the value $\sigma = 3.0$ is too high, leading to an overly low acceptance rate (0.2%), a poor estimate (1.000) of the mean functional value with large standard error (0.083) and large root mean squared error (0.085). On the other hand, the value $\sigma = 0.7$ is just right, leading to a nearly-optimal acceptance rate (23.0%), a good estimate (1.032) of the mean functional value with smaller standard error (0.019) and smaller root mean squared error (0.085).

This confirms that, when scaling the increment covariance as σI_d, it is optimal to find σ to make the acceptance rate close to 0.234.

2.5.2. Inhomogeneous Covariance.

To consider the effect of non-diagonal proposal increments, we again consider a case where the target density π is that of $N(0, \Sigma)$, again in dimension $d = 10$, but now we take $\Sigma = \mathrm{diag}(1^2, 2^2, 3^2, \ldots, 10^2)$. Thus, the individual covariances are now highly variable. Since the last coordinate now has the highest variance and is thus most "interesting", we consider the functional given by the square of the last coordinate. So, the functional's true mean is now 100. We again start the algorithms with the initial value $X_0 = (1, 0, 0, \ldots, 0)$.

We first consider a usual RWM algorithm, with proposal increment distribution $N(0, \sigma^2 I_d)$, with $\sigma = 0.7$ chosen to get an acceptance rate close to the optimal value of 0.234. The result (again upon running the algorithm for 100,000 iterations, repeated 10 times to compute a sample standard error) is as follows:

σ	mean acc. rate	estimate	RMSE
0.7	0.230	114.8 ± 28.2	30.5

We thus see that, even though σ was well chosen, the resulting algorithm still converges poorly leading to a poor estimate (114.8) with large standard error (28.2) and large RMSE (30.5).

Next we consider running the modified algorithm where now the increment proposal is equal to $N(0, \sigma^2 \Sigma)$ where Σ is the target covariance matrix as above, but otherwise the run is identical. In this case, we find the following:

σ	mean acc. rate	estimate	RMSE
0.7	0.294	100.25 ± 1.91	1.83

Comparing the two tables, we can see that the improvement from using an increment proposal covariance proportional to the target covariance (rather than the identity matrix) is very dramatic. The estimate (100.25) is much closer to the true value (100), with much smaller standard error (1.91) and much smaller RMSE (1.83). (Furthermore, the second simulation was simply run with $\sigma = 0.7$ as in the first simulation, leading to slightly too large an acceptance rate, so a slightly larger σ would make it even better.) This confirms, as shown by Roberts and Rosenthal (2001), that when running a Metropolis algorithm, it is much better to use increment proposals which mimic the covariance of the target distribution if at all possible.

Of course, in general the target covariance matrix will not be known, and it is not at all clear (especially in high dimensions) how one could arrange for proposal increment covariances to mimic the target covariance. One promising solution is adaptive MCMC, discussed in the next section. In particular, Section 3 considers the adaptive Metropolis algorithm and shows how it can successfully mimic the target covariance without any *a priori* knowledge about it, even in hundreds of dimensions.

2.6. Frequently Asked Questions.

Isn't a larger acceptance rate always preferable?

No. For RWM, if the acceptance rate is close to 1, this means the proposal increments are so small that the algorithm is highly inefficient despite all the acceptances.

Is it essential that the acceptance rate be exactly 0.234?

No. As shown in Figure 5, the algorithm's efficiency remains high whenever the acceptance rate is between about 0.1 and 0.6.

Are these asymptotic results relevant to finite-dimensional problems?

Yes. While the theorems are only proven as $d \to \infty$, it appears that in many cases the asymptotics approximately apply whenever $d \geq 5$, so the infinite-dimensional results are good approximations to finite-dimensional situations.

Do these results hold for all target distributions?

No. They are only proved for very special cases involving independent target components. However, within that class they appear to be fairly robust (albeit sometimes with an even *lower* optimal acceptance rate than 0.234), and simulations seem to suggest that they approximately hold in other cases too. Furthermore, by change-of-basis, the results apply to all normal target distributions, too. And, the general principle that the scaling should be neither too large nor too small, applies much more generally, to virtually all "local" MCMC algorithms.

Do these results hold for multi-modal distributions?

In principle, yes, at least for distributions with independent (though perhaps multi-modal) components. However, the asymptotic acceptance rate is by definition the acceptance rate with respect to the *entire* target distribution. So, if a sampler is stuck in just one mode, it may mis-represent the asymptotic acceptance rate, leading to an incorrect estimate of the asymptotic acceptance rate, and a mis-application of the theorem.

In high dimensions, is the proposal scaling parameter σ the only quantity of interest?

No. The entire proposal distribution is of interest. In particular, it is best if the covariance of the proposal increment distribution mimics the covariance of the target distribution as much as possible. However, often significant gains can be realised simply by optimising σ according to the theorems.

Doesn't optimality depend on which criterion is used?

Yes in general, but these asymptotic diffusion results are valid for *any* optimality measure. That is because in the limit the processes each represent precisely the same *diffusion*, just scaled with a different speed factor. So, running a suboptimal algorithm for n steps is precisely equivalent (in the limit) to running the optimal algorithm for m steps, where $m < n$. In other words, with a suboptimal algorithm you have to run for longer to achieve precisely the same result, which is less efficient by any sensible efficiency measure at all, including all of those in Subsection 1.4.

Do these results hold for, say, Metropolis-within-Gibbs algorithms?

No, since they are proved for full-dimensional Metropolis updates only. Indeed, the Metropolis-within-Gibbs algorithm involves updating just one coordinate at a time, and thus essentially corresponds to the case $d = 1$. In that case, it appears that the optimal acceptance rate is usually closer to 0.44 than 0.234.

Isn't it too restrictive to scale σ specifically as $O(d^{-1/2})$ for RWM, or $O(d^{-1/6})$ for MALA? Wouldn't other scalings lead to other optimality results?

No, a smaller scaling would correspond to letting $\ell \to 0$, while a larger scaling would correspond to letting $\ell \to \infty$, either of which would lead to an asymptotically zero-efficiency algorithm (cf. Figure 5). The $O(d^{-1/2})$ or $O(d^{-1/6})$ scaling is the only one that leads to a non-zero limit, and are thus the only scaling leading to optimality as $d \to \infty$.

3. Adaptive MCMC.

Even if we have some idea of what criteria makes an MCMC algorithm optimal, this still leaves the question of how to *find* this optimum, i.e. how to run a Markov chain with (approximately) optimal characteristics. For example, even if we are convinced that an acceptance rate of 0.234 is optimal, how do we find the appropriate proposal scaling to achieve this?

One method, commonly used, is by trial-and-error: if the acceptance rate seems too high, then we reduce the proposal scaling σ and try again (or if it seems too low, then we increase the scaling). This method is often successful, but it is generally time-consuming, requiring repeated manual intervention by the user. Furthermore, such a method cannot hope to find more complicated improvements, for example making the proposal covariance matrix Σ_p approximately proportional to the (unknown) target covariance matrix Σ as in (5) (which requires choosing $d(d-1)/2$ separate covariance matrix entries). It is possible to use more refined versions of this, for example with increasing trial run lengths to efficiently zero in on good proposal scale and shape values (Pasarica and Gelman, 2006), but this is still not sufficient in difficult high-dimensional problems.

As an alternative, we consider algorithms which themselves try to improve the Markov chain. Specifically, let $\{P_\gamma\}_{\gamma\in\mathcal{Y}}$ be a family of Markov chain kernels, each having the same stationary distribution π. Let Γ_n be the chosen kernel choice at the n^{th} iteration, so

$$\mathbf{P}(X_{n+1} \in A \,|\, X_n = x, \Gamma_n = \gamma, X_{n-1}, \ldots, X_0, \Gamma_{n-1}, \ldots, \Gamma_0) \;=\; P_\gamma(x, A)$$

for $n = 0, 1, 2, \ldots$. Here the $\{\Gamma_n\}$ are updated according to some adaptive updating algorithm. In principle, the choice of Γ_n could depend on the entire history $X_{n-1}, \ldots, X_0, \Gamma_{n-1}, \ldots, \Gamma_0$, though in practice it is often the case that the pairs process $\{(X_n, \Gamma_n)\}_{n=0}^{\infty}$ is Markovian. In general the algorithms are quite easy to implement, requiring only moderate amounts of extra computer programming (and there are even some efforts at generic adaptive software, e.g. Rosenthal, 2007).

Whether such an adaptive scheme will improve convergence depends, obviously, on the adaptive algorithm selected. An even more fundamental question, which we now consider, is whether the adaption might *destroy* convergence.

3.1. Ergodicity of Adaptive MCMC.

One might think that, as long as each individual Markov chain P_γ converges to π, therefore any adaptive mixture of the chains must also converge to π. However, this is not the case. For a simple counter-example (illustrated interactively by Rosenthal, 2004; see also Atchadé and Rosenthal, 2005, and Roberts and Rosenthal, 2005), let $\mathcal{Y} = \{1, 2\}$, let $\mathcal{X} = \{1, 2, 3, 4\}$, let $\pi(1) = \pi(3) = \pi(4) = 0.333$ and $\pi(2) = 0.001$. Let each P_γ be a RWM algorithm, with proposal $Y_{n+1} \sim \text{Uniform}\{X_n - 1, X_n + 1\}$ for P_1, or $Y_{n+1} \sim \text{Uniform}\{X_n - 2, X_n - 1, X_n + 1, X_n + 2\}$ for P_2. (Of course, any proposed moves out of $\mathcal{X}$ are always rejected, i.e. $\pi(x) = 0$ for $x \notin \mathcal{X}$.) Define the adaption by saying that $\Gamma_{n+1} = 2$ if the n^{th} proposal was accepted, otherwise $\Gamma_{n+1} = 1$. Then each P_γ is reversible with respect to π. However, the adaptive algorithm can get "stuck" with $X_n = \Gamma_n = 1$ for long stretches (and only escape with probability 0.001/0.333), so the limiting distribution of X_n is weighted too heavily towards 1 (and too lightly towards 3 and 4).

In light of such counter-examples, it is important to have sufficient conditions to guarantee convergence in distribution of $\{X_n\}$ to π. In recent years, a number of authors (Haario et al., 2001; Atchadé and Rosenthal, 2005; Andrieu and Moulines, 2006; Roberts and Rosenthal, 2007; Giordani and Kohn, 2006; Andrieu and Atchadé, 2007) have proved ergodicity of adaptive MCMC under various assumptions.

In particular, Roberts and Rosenthal (2005) proved that $\lim_{n\to\infty} \sup_{A\subseteq\mathcal{X}} \|\mathbf{P}(X_n \in A) - \pi(A)\| = 0$ (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* (a.k.a. *Vanishing*) *Adaptation* condition

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

and also the *Containment* (a.k.a. *Bounded Convergence*) condition

$$\{M_\epsilon(X_n, \Gamma_n)\}_{n=0}^{\infty} \quad \text{is bounded in probability}\,, \qquad \epsilon > 0\,, \tag{7}$$

where $M_\epsilon(x,\gamma) = \inf\{n \geq 1 : \|P_\gamma^n(x,\cdot) - \pi(\cdot)\| \leq \epsilon\}$ is the convergence time of the kernel P_γ when beginning in state $x \in \mathcal{X}$.

Now, (7) is a technical condition which is satisfied for virtually all reasonable adaptive schemes. For example, it holds 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. It also holds for adaptive RWM and Metropolis-within-Gibbs algorithms under very general conditions (Bai et al., 2008). (It is, however, possible to construct pathalogical counter-examples, where containment does not hold; see Yang, 2008b, and Bai et al., 2008.) So, in practice, the requirement (7) can be largely ignored.

By contrast, condition (6) is more fundamental. It requires that the amount of adapting at the n^{th} iteration goes to 0 as $n \to \infty$. (Note that the *sum* of the adaptions can still be infinite, i.e. an infinite total amount of adaption is still permissible, and it is not necessarily required that the adaptive parameters $\{\Gamma_n\}$ converge to some fixed value.) Since the user can choose the adaptive updating scheme, (6) can be ensured directly through careful planning. For example, if the algorithm adapts at the n^{th} iteration only with probability $p(n)$, then (6) is automatically satisfied if $p(n) \to 0$. Alternatively, if the choice of γ depends on an empirical average over iterations 1 through n, then the influence of the n^{th} iteration is just $O(1/n)$ and hence goes to 0.

Such results allow us to update our parameters $\{\Gamma_n\}$ in virtually any manner we wish, so long as (6) holds. So, what adaptions are beneficial?

3.2. Adaptive Metropolis (AM).

The first important modern use of adaptive MCMC was the Adaptive Metropolis (AM) algorithm of Haario et al. (2001). This algorithm is motivated by the observation (5) that for RWM in $\mathbf{R}^d$, at least with normal target distributions, it is optimal to have a proposal covariance matrix of the form $(2.38)^2/d$ times the target covariance matrix Σ. Since Σ is in general unknown, it is estimated by Σ_n, the empirical covariance matrix of $X_0, \ldots, X_n$.

Thus, the AM algorithm essentially uses a proposal distribution for the n^{th} iteration given by

$$Y_{n+1} \sim N(X_n, [(2.38)^2/d] \Sigma_n) .$$

To ensure that the proposal covariances don't simply collapse to 0 (which could violate (7)), Haario et al. (2001) added ϵI_d to Σ_n at each iteration, for some small $\epsilon > 0$. Another possibility (Roberts and Rosenthal, 2006) is to instead let the proposal be a mixture distribution of the form

$$(1 - \beta) N(X_n, [(2.38)^2/d] \Sigma_n) + \beta N(X_n, \Sigma_0)$$

for some $0 < \beta < 1$ and some fixed non-singular matrix Σ_0 (e.g., $\Sigma_0 = [(0.1)^2/d] I_d$). (With either version, it is necessary to use some alternative fixed proposal distribution for the first few iterations when the empirical covariance Σ_n is not yet well-defined.)

Since empirical estimates change at the n^{th} iteration by only $O(1/n)$, it follows that the Diminishing Adaptation condition (6) will be satisfied. Furthermore, the containment condition (7) will certainly be satisfied if one restricts to compact regions (Haario et al., 2001; Roberts and Rosenthal, 2006), and in fact containment still holds provided the target density π decays at least polynomially in each coordinate, a very mild assumption (Bai et al., 2008). So, AM is indeed a valid sampling algorithm.

Computer simulations (Roberts and Rosenthal, 2006) demonstrate that this AM algorithm will indeed "learn" the target covariance matrix, and approach an optimal algorithm, even in very high dimensions. While it may take many iterations before the adaption significantly improves the algorithm, in the end it will converge enormously faster than a non-adapted RWM algorithm. The following figures show a trace plot of the first coordinate (Figure 6), and also a graph of the inhomogeneity factor b in (4) (Figure 7), for an AM run in dimension $d = 100$ (where the target was a normal distribution with an irregular and highly skewed covariance matrix). They show that the run initially underestimates the variability of the first coordinate, which would lead to drastically incorrect estimates. However, after about 250,000 iterations, the algorithm has "found" a good proposal increment covariance matrix, so that b gets close to 1, and the trace plot correctly finds the true variability of the first coordinate. Such adaptation could never have been done manually, because of the large dimension, but the computer eventually finds a good algorithm. This shows the potential of adaptive MCMC to find good algorithms that cannot be found by hand.

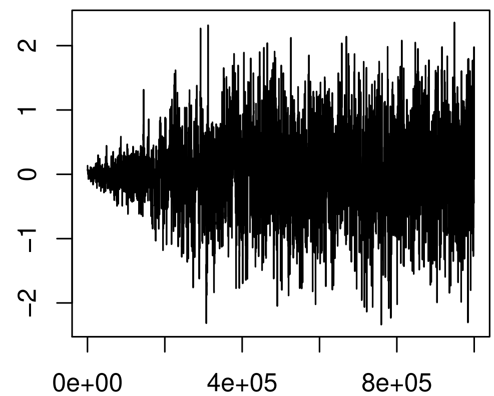

Figure 6. Trace plot of first coordinate of AM in dimension 100.

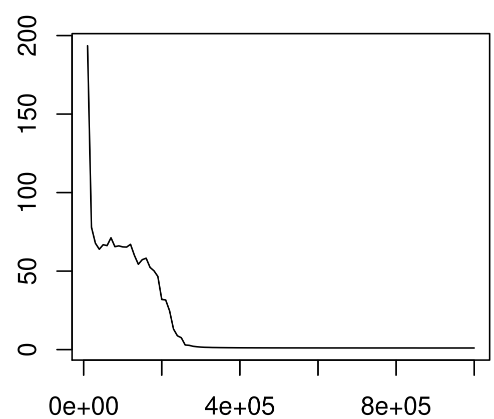

Figure 7. Trace plot of inhomogeneity factor b for AM in dimension 100.

3.3. Adaptive Metropolis-Within-Gibbs.

A standard alternative to the usual full-dimensional Metropolis algorithm is the "Metropolis-Within-Gibbs" algorithm (arguably a misnomer, since the original work of Metropolis et al., 1953, corresponded to what we now call Metropolis-Within-Gibbs). Here the variables are updated one at a time (in either systematic or random order), each using a Metropolis algorithm with a one-dimensional proposal.

To be specific, suppose that the i^{th} coordinate is updated using a proposal increment distribution $N(0, e^{2\,ls_i})$, so ls_i is the log of the standard deviation of the increment. Obviously, we would like to find optimal values of the ls_i, which may of course be different for the different variables. We even have a rule of thumb from the end of Subsection 2.3, that each ls_i should be chosen so that the acceptance rate is approximately 0.44. However, even with this information, it is very difficult (if not impossible) in high dimensions to optimise each ls_i manually. Instead, an adaptive algorithm might be used.

One way (Roberts and Rosenthal, 2006) to adapt the ls_i values is to break up the run into "batches" of, say, 50 iterations each. After the n^{th} batch, 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 as close as possible to 0.44. 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. (A related component-wise adaptive scaling method, a one-dimensional analog of the original Adaptive Metropolis algorithm of Haario et al., 2001, is presented in Haario et al., 2005.)

To satisfy condition (6) we require $\delta(n) \to 0$; for example, we might take $\delta(n) = \min(0.01, n^{-1/2})$. As for (7), it is easily seen to be satisfied if we restrict each ls_i to a finite interval $[-M, M]$. However, even this is not necessary, since it is proved by Bai et al. (2008) that (7) is always satisfied for this algorithm, provided only that the target density π decreases at least polynomially in each direction (a very mild condition). Hence, the restriction (7) is once again not of practical concern.

Simulations (Roberts and Rosenthal, 2006) indicate that this adaptive Metropolis-within-Gibbs algorithm does a good job of correctly scaling the ls_i values, even in dimensions as high as 500, leading to chains which mix much faster than those with pre-chosen proposal scalings. The algorithm has recently been applied successfully to high-dimensional inference for statistical genetics (Turro et al., 2007). We believe it will be applied to many more sampling problems in the near future. Preliminary general-purpose software to implement this algorithm is now available (Rosenthal, 2007).

3.4. State-Dependent Proposal Scalings.

Another approach involves letting the proposal scaling depend on the current state X_n, so that e.g. given $X_n = x$, we might propose $Y_{n+1} \sim N(x, \sigma_x^2)$. In this case, the acceptance probability (1) becomes

$$\alpha(x,y) \;=\; \min\Big[1,\; \frac{\pi(y)}{\pi(x)}\;(\sigma_x/\sigma_y)^d\;\exp\big(-\frac{1}{2}(x-y)^2(\sigma_y^{-2}-\sigma_x^{-2})\big)\Big]\,. \tag{8}$$

The functional form of σ_x can be chosen and adapted in various ways to attempt to achieve efficient convergence.

For example, in many problems the target distribution becomes more spread out as we move farther from the origin. In that case, it might be appropriate to let, say, $\sigma_x = e^a(1+|x|)^b$ where a and b are determined adaptively. For example, we could again divide the run into batches of 50 iterations as in the previous subsection. After each iteration, the algorithm updates a by adding or subtracting $\delta(n)$ in an effort to make the acceptance rate as close as possible to e.g. 0.234 or 0.44. The algorithm also adds or subtracts $\delta(n)$ to b in an effort to equalise the acceptance rates in the two regions $\{x \in \mathcal{X} : |x| > C\}$ and $\{x \in \mathcal{X} : |x| \le C\}$ for some fixed C.

Once again, condition (6) is satisfied provided $\delta(n) \to 0$, and (7) is satisfied under very mild conditions. So, this provides a convenient way to give a useful functional form to σ_x, without knowing in advance what values of a and b might be appropriate. Simulations (Roberts and Rosenthal, 2006) indicate that this adaptive algorithm works well, at least in simple examples.

Another approach, sometimes called the Regional Adaptive Metropolis Algorithm (RAMA), use a finite partition of the state space: $\mathcal{X} = \mathcal{X}_1 \dot{\cup} \ldots \dot{\cup} \mathcal{X}_m$. The proposal scaling is then given by $\sigma_x = e^{a_i}$ whenever $x \in \mathcal{X}_i$, with the acceptance probability (8) computed accordingly. Each of the values a_i is again adapted after each batch of iterations, by adding or subtracting $\delta(n)$ in an attempt to make the acceptance fraction of proposals from $\mathcal{X}_i$ close to 0.234. (As a special case, if there were no visits to $\mathcal{X}_i$ during the batch, then we always *add* $\delta(n)$ to a_i, to avoid the problem of a_i becoming so low that proposed moves to $\mathcal{X}_i$ are never accepted.) Once again, the algorithm will be valid under very mild conditions provided $\delta(n) \to 0$.

Recent work of Craiu et al. (2008) considers certain modifications of RAMA, in which multiple copies of the algorithm are run simultaneously in an effort to be sure to "learn" about *all* modes rather than getting stuck in a single mode. Their work also allows the proposal distribution to be a weighted mixture of the different $N(x, e^{2a_i})$, to allow for the possibility that the partition $\{\mathcal{X}_i\}$ was imperfectly chosen. It appears that such greater flexibility will allow for wider applicability of RAMA-type algorithms.

Of course, Langevin (MALA) algorithms may also be regarded as a type of state-dependent scaling, and it is possible to study adaptive versions of MALA as well (Atchadé, 2006).

3.5. Limit Theorems.

Many applications of MCMC make use of such Markov chain limit theorems as the Weak Law of Large Numbers (WLLN), Strong Law of Large Numbers (SLLN), and Central Limit Theorem (CLT), in order to guarantee good asymptotic estimates and estimate standard errors (see e.g. Tierney, 1994; Jones and Hobert, 2001; Hobert et al., 2002; Roberts and Rosenthal, 2004; and Jones, 2004). So, it is natural to ask if such limit theorems hold for adaptive MCMC as well.

Under the assumptions of Diminishing Adaptation and Containment, the WLLN does hold for all *bounded* functionals (Roberts and Rosenthal, 2007, Theorem 23). So, this at least means that when using adaptive MCMC for estimating means of bounded functionals, one will obtain an accurate answer with high probability if the run is sufficiently long.

For unbounded functionals, the WLLN *usually* still holds, but not always (Yang, 2008a, Theorem 2.1). Even for bounded functionals, the SLLN may not hold (Roberts and Rosenthal, 2007, Example 24), and that same example shows that a CLT might not hold as well. So, this suggests that the usual estimation of MCMC standard errors may be more challenging for adaptive MCMC if we assume only Diminishing Adaptation and Containment.

Under stronger assumptions, more can be said. For example, Andrieu and Moulines (2006; see also Andrieu and Atchadé, 2007, and Atchadé, 2007) prove various limit theorems (including CLTs) for adaptive MCMC algorithms, assuming that the adaptive parameters converge to fixed values sufficiently quickly. They also prove that such adaptive algorithms will inherit many of the asymptotic optimality properties of the corresponding fixed-parameter algorithms. Such results facilitate further applications of adaptive MCMC, however they require various technical conditions which may be difficult to check in practice.

3.6. Frequently Asked Questions.

Can't I adapt my MCMC algorithm any way I like, and still preserve convergence?

No. In particular, if the Diminishing Adaption condition (6) does not hold, then there are simple counter-examples showing that adaptive MCMC can converge to the wrong answer, even though each individual Markov chain kernel would correctly converge to π.

Do I have to learn lots of technical conditions before I can apply adaptive MCMC?

Not really. As long as you satisfy Diminishing Adaption (6), which is important but quite intuitive, then your algorithm will probably be asymptotically valid.

Have adaptive MCMC algorithms actually been used to speed up convergence on high-dimensional problems?

Yes, they have. Simulations on test problems involving hundreds of dimensions have been quite successful (Roberts and Rosenthal, 2006), and adaptive Metropolis-within-Gibbs has also been used on statistical genetics problems (Turro et al., 2007).

Does adaption have to be designed specifically to seek out optimal parameter values?

No. The ergodicity results presented herein do *not* require that the parameters $\{\Gamma_n\}$ converge at all, only that they satisfy (6) which still allows for the possibility of infinite total adaption. However, many of the specific adaptive MCMC algorithms proposed are indeed designed to attempt to converge to specific values (e.g., to proposal scalings which give an asymptotic acceptance rate of 0.234).

Why not just do the adaption by hand, with trial runs to determine optimal parameter values, and then a long run using these values?

Well, if you can really determine optimal parameter values from a few trial runs, then that's fine. However, in high dimensions, with many parameters to choose (e.g., a large proposal covariance matrix), it is doubtful that you can find good parameter values manually.

Suppose I just have the computer adapt for some fixed, finite amount of time, and then continue the run without further adapting. Won't that guarantee asymptotic convergence to π?

Yes, it will (provided each individual kernel P_γ is ergodic), and this is a sensible method to try. However, it may be unclear how much adaption should be done before you stop. For example, with adaptive Metropolis in 200 dimensions, it took well over a million iterations (Roberts and Rosenthal, 2006) before a truly good proposal covariance matrix was found – and it was not clear *a priori* that it would take nearly so long.

Can I use adaption for other types of MCMC algorithms, like the Gibbs sampler?

In principle, yes. For example, an adaptive Gibbs sampler could adapt such quantities as the order of update of coordinates (for systematic-scan), or the probability weights of various coordinates (for random-scan), or coordinate blockings for joint updates, or such reparameterisations as rotations and centerings and so on. Only time will tell what adaptions turn out to be useful in what contexts.

Am I restricted to the specific adaptive MCMC algorithms (Adaptive Metropolis, Adaptive Metropolis-within-Gibbs, RAMA, . . .) presented herein?

Not at all! You can make up virtually any rules for how your Markov chain parameters $\{\Gamma_n\}$ adapt over time, as long as the adaption diminishes, and your algorithm will probably be valid. The challenge is then to find sensible/clever adaption rules. Hopefully more and better adaptive methods will be found in the future!

Are any other methods, besides adaptive MCMC, available to help algorithms "learn" how to converge well?

Yes, there are many. For example, particle filters (e.g. Pitt and Sheppard, 1999), population Monte Carlo (e.g. Cappé et al., 2004), and sequential Monte Carlo (e.g. Del Moral et al., 2006), can all be considered as methods which attempt to "learn" faster convergence as they go. However, the details of their implementations are rather different than the adaptive MCMC algorithms presented herein.

4. Conclusion.

We have reviewed optimal proposal scaling results, and adaptive MCMC algorithms.

While the optimal scaling theorems are all proved under very restrictive and unrealistic assumptions (e.g., target distributions with independent coordinates), they appear to provide useful guidelines much more generally. In particular, results about asymptotic acceptance rates provide useful benchmarks for Meteropolis algorithms in a wide variety of settings.

Adaptive MCMC algorithms appear to provide simple, intuitive methods of finding quickly-converging Markov chains without great effort on the part of the user (aside from the initial programming, and there is even some generic software available, e.g. Rosenthal, 2007). While certain conditions (notably Diminishing Adaptation) must be satisfied to guarantee asymptotic convergence, these conditions are generally not onerous or difficult to achieve.

Overall, we feel that these results indicate the widespread applicability of both optimal scaling and adaptive MCMC algorithms to many different MCMC settings (Roberts and Rosenthal, 2006; Turro et al., 2007), including to complicated high-dimensional distributions. We hope that many MCMC users will be guided by optimal scaling results, and experiment with adaptive algorithms, in their future applications.

REFERENCES

C. Andrieu and Y.F. Atchadé (2007), On the efficiency of adaptive MCMC algorithms. Elec. Comm. Prob. **12**, 336–349.

C. Andrieu and E. Moulines (2006), On the ergodicity properties of some adaptive Markov Chain Monte Carlo algorithms. Ann. Appl. Prob. **16**, 1462–1505.

Y.F. Atchadé (2006), An adaptive version for the Metropolis adjusted Langevin algorithm with a truncated drift. Meth. Comp. Appl. Prob. **8**, 235-254.

Y.F. Atchadé (2007), A cautionary tale on the efficiency of some adaptive Monte Carlo schemes. 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. In preparation.

M. Bédard (2008), Efficient Sampling using Metropolis Algorithms: Applications of Optimal Scaling Results. J. Comp. Graph. Stat. **17**, 1–21.

M. Bédard (2006), Optimal Acceptance Rates for Metropolis Algorithms: Moving Beyond 0.234. Stoch. Proc. Appl., to appear.

M. Bédard (2007), Weak Convergence of Metropolis Algorithms for Non-iid Target Distributions. Ann. Appl. Prob. **17**, 1222–44.

M. Bédard and J.S. Rosenthal (2008), Optimal Scaling of Metropolis Algorithms: Heading Towards General Target Distributions. Can. J. Stat., to appear.

O. Cappé, A. Guillin, J.M. Marin, and C.P. Robert (2004) Population Monte Carlo. J. Comp. Graph. Stat. **13**, 907–930.

R.V. Craiu, J.S. Rosenthal, and C. Yang (2008), Learn from thy neighbor: Parallel-Chain Adaptive MCMC. In preparation.

P. Del Moral, A. Doucet, and A. Jasra (2006), Sequential Monte Carlo samplers. J. Roy. Stat. Soc. Ser. B **68**, 411–436.

A. Gelman, G.O. Roberts, and W.R. Gilks (1996), Efficient Metropolis jumping rules. In *Bayesian Statistics 5*, ed. J. Bernardo et al., 599–607. Oxford University Press.

P. Giordani and R. Kohn (2006), Adaptive independent Metropolis-Hastings by fast estimation of mixtures of normals. Preprint.

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.

J.P. Hobert, G.L. Jones, B. Presnell, and J.S. Rosenthal (2002), On the Applicability of Regenerative Simulation in Markov Chain Monte Carlo. Biometrika **89**, 731–743.

G.L. Jones (2004), On the Markov chain central limit theorem. Prob. Surv. **1** (2004), 299–320.

G.L. Jones and J.P. Hobert (2001), Honest exploration of intractable probability distributions via Markov chain Monte Carlo. Stat. Sci. **16**, 312–334.

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.

A. Mira (2001), Ordering and improving the performance of Monte Carlo Markov chains. Stat. Sci. **16**, 340–350.

P.J. Neal, G.O. Roberts, and W.K. Yuen (2007), Optimal Scaling of Random Walk Metropolis algorithms with discontinuous target densities. Preprint.

C. Pasarica and A. Gelman (2010), Adaptively scaling the Metropolis algorithm using expected squared jumped distance. Statistica Sinica **20**, to appear.

M.K. Pitt and N. Shephard (1999), Filtering via simulation: auxiliary particle filters. J. Amer. Stat. Assoc. **94**, 1403–1412.

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 (2004), General state space Markov chains and MCMC algorithms. Prob. Surv. **1**, 20–71.

G.O. Roberts and J.S. Rosenthal (2006), Examples of Adaptive MCMC. Preprint.

G.O. Roberts and J.S. Rosenthal (2007), Coupling and Ergodicity of Adaptive MCMC. J. Appl. Prob. **44**, 458–475.

G.O. Roberts and R.L. Tweedie (1996), Exponential convergence of Langevin diffusions and their discrete approximations. Bernoulli **2**, 341-363.

G.O. Roberts and R.L. Tweedie (1999), Bounds on regeneration times and convergence rates for Markov chains. Stoch. Proc. Appl. **80**, 211–229. See also the corrigendum, Stoch. Proc. Appl. **91** (2001), 337–338.

J.S. Rosenthal (1995), Minorization conditions and convergence rates for Markov chain Monte Carlo. J. Amer. Stat. Assoc. **90**, 558–566.

J.S. Rosenthal (2002), Quantitative convergence rates of Markov chains: A simple account. Elec. Comm. Prob. **7**, No. 13, 123–128.

J.S. Rosenthal (2004), Adaptive MCMC Java Applet. Available at:
http://probability.ca/jeff/java/adapt.html

J.S. Rosenthal (2007), AMCMC: An R interface for adaptive MCMC. Comp. Stat. Data Anal. **51**, 5467–5470. [Related software available at probability.ca/amcmc.]

C. Sherlock (2006), Methodology for Inference on the Markov Modulated Poisson Processes and Theory for Optimal Scaling of the Random Walk Metropolis. Ph.D. dissertation, Lancaster University.

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

C. Yang (2008a), On The Weak Law of Large Numbers for Unbounded Functionals for Adaptive MCMC. Preprint.

C. Yang (2008b), Recurrent and Ergodic Properties of Adaptive MCMC. Preprint.