Exploring an Adaptive Metropolis Algorithm

Benjamin Shaby

Department of Statistical Science

Duke University

Durham, NC 27708

Martin T. Wells

Department of Statistical Science

Cornell University

Ithaca, NY 14853

December 23, 2010

Abstract

While adaptive methods for MCMC are under active development, their utility has been under-recognized. We briefly review some theoretical results relevant to adaptive MCMC. We then suggest a very simple and effective algorithm to adapt proposal densities for random walk Metropolis and Metropolis adjusted Langevin algorithms. The benefits of this algorithm are immediate, and we demonstrate its power by comparing its performance to that of three commonly-used MCMC algorithms that are widely-believed to be extremely efficient. Compared to data augmentation for probit models, slice sampling for geostatistical models, and Gibbs sampling with adaptive rejection sampling, the adaptive random walk Metropolis algorithm that we suggest is both more efficient and more flexible.

KEYWORDS: adaptive, Markov chain Monte Carlo, Metropolis algorithm, Non-conjugate priors

1 Introduction

Markov Chain Monte Carlo sampling has by now gained wide recognition as being an essential tool for carrying out many Bayesian analyses. One of the most popular, flexible, as well as oldest, MCMC algorithms is Metropolis Hastings (MH) (Metropolis et al. 1953; Hastings 1970). For all its ubiquity, a generally under-appreciated aspect of MH algorithms is how widely their performance can vary due to different choices of tuning parameters.

Tuning MH usually proceeds by running short pilot chains and manually adjusting a single parameter to achieve a reasonable acceptance rate. Aside from being tedious and ad hoc, this approach has the obvious limitation of only allowing practical optimization of a single parameter, when often more flexibility is desirable.

One particular class of Metropolis Hastings algorithm is the Metropolis Adjusted Langevin Algorithm (MALA) (Besag 1994; Roberts and Tweedie 1996). Inspired by stochastic models of molecular dynamics, MALA works, informally, by encouraging the sampling process to move "uphill" towards regions of higher probability mass. The resulting Markov chain has many desirable properties relative to Metropolis algorithms, the most important of which is markedly faster mixing.

Langevin samplers, however, also possess two undesirable properties that can cause difficulties in practice. The first is that they are extremely sensitive to the choice of tuning parameter, where small changes in the tuning parameter can mean the difference between a well-mixing chain and a chain that does not jump at all. The second is that MALA chains behave very differently in transience and stationarity, so that a good choice of tuning parameter in transience can be an extremely poor choice in stationarity.

In tuning MALA chains then, one is left in a conundrum: the many pilot runs required to obtain a tuning parameter within the tiny usable window produce, necessarily, a sub-optimal chain in stationarity. That is, pilot runs, because they are pilot runs, can, at best, produce tuning parameters that are optimal for the transient phase. This tuning will be sub-optimal for the stationary phase, which is ultimately what one cares about. On the other hand, if one knew the optimal tuning for stationarity, the chain so tuned may behave so poorly in transience that it never enters a stationary phase.

This paper explores an adaptive tuning strategy for MH samplers, including MALA. The algorithm we describe in Section 3, which we call Log-Adaptive Proposals (LAP), quickly produces samplers with excellent performance characteristics, essentially without requiring human intervention. It painlessly generates parameter tuning that is at once more flexible and more accurate than is possible with manual tuning. More subtly, it is capable of constructing MALA samplers that mix well in transience, and then automatically adjust themselves for optimality in stationarity.

The automatic tuning of LAP is so effective, in fact, that it makes Metropolis samplers at least as efficient as, and in some cases substantially more efficient than, specialized MCMC algorithms designed to sample from posteriors arising from specific classes of models. Its wide applicability, compatibility with almost any priors, simplicity of implementation, and excellent performance make MH with LAP an attractive option in almost any context in which MCMC is required.

Of course, if computational time and resources are not a concern, then using efficient samplers is not particularly important. However, with increasing interest in hierarchical models for high dimensional data common in areas such as microarray (e.g. Gottardo et al. 2006; Nott et al. 2007) and spatial analysis (e.g. Christensen and Waagepetersen 2002; Royle and Wikle 2005; Banerjee et al. 2008), computational efficiency is critical. In high-dimensional settings like these, each likelihood evaluation, and thus each MCMC iteration, is extremely computationally expensive, and therefore it is highly desirable to perform as few iterations as possible. This is accomplished by using efficient MCMC samplers.

The remainder of this paper is organized as follows. In Section 2 we review some important theoretical results that suggest why tuning MH and MALA samplers is so important for efficiency. In Section 3 we describe a simple and effective automatic tuning algorithm and demonstrate its use with MH and MALA samplers. Finally, in Section 4 we compare the efficiency of this algorithm to well-known specialized algorithms for sampling from posterior distributions, concluding that the automatically-tuned MH sampler performs at least as well as its competitors, and in come cases dramatically better.

2 A review of Some Scaling Results

Throughout this paper we shall consider a d-dimensional *target* distribution Π with un-normalized density (with respect to the Lebesgue measure) denoted by π. It is this target distribution from which we wish to sample. Most commonly, in Bayesian data analysis, π is the joint posterior density of the parameters given the observed data.

The Metropolis Hastings algorithm requires a proposal kernel $Q(\mathbf{x}, \cdot)$, with density (with respect to the Lebesgue measure) $q(\mathbf{x}, \cdot)$, and proceeds according to Algorithm 1.

The collection $\{\mathbf{X}_t, t = 1, \ldots\}$ are then samples from a Markov chain with stationary distribution Π. Whenever the proposal $\mathbf{X}^*$ is accepted, we say the chain has "jumped".

In this section we restrict our focus to two special cases of MH samplers: the Random Walk Metropolis Algorithm (RWM) with Gaussian proposals and the Metropolis Adjusted Langevin Algorithm, also with Gaussian proposals. These two algorithms differ only in their proposal distributions. For RWM, $Q(\mathbf{x}, \cdot) \sim N(\mathbf{x}, \sigma_m \mathbf{\Sigma}_0)$ for some positive scaling constant σ_m and some $d \times d$ positive definite matrix $\mathbf{\Sigma}_0$. In this case, the ratio (1) reduces to $r(\mathbf{X}, \mathbf{X}^*) =$

Algorithm 1 Metropolis Hastings algorithm

```
Require: Set initial values for the vector $\mathbf{X}_{(0)}$.
 1: for $t = 0$ to $T$ do
 2:     Draw a proposal value $\mathbf{X}^*$ from the density $q(\mathbf{X}_{(t)}, \cdot)$.
 3:     Calculate the ratio
            $$r(\mathbf{X}, \mathbf{X}^*) = \frac{\pi(\mathbf{X}^*)q(\mathbf{X}^*, \mathbf{X})}{\pi(\mathbf{X})q(\mathbf{X}, \mathbf{X}^*)}. \quad (1)$$
 4:     Draw $Y \sim U(0, 1)$
 5:     if $r(\mathbf{X}, \mathbf{X}^*) > Y$ then
 6:         Set $\mathbf{X}_{(t+1)} \leftarrow \mathbf{X}^*$
 7:     else
 8:         Set $\mathbf{X}_{(t+1)} \leftarrow \mathbf{X}_t$
 9:     end if
10: end for
```

$\frac{\pi(\mathbf{X}^*)}{\pi(\mathbf{X})}$ because the proposal density is symmetric in its arguments.

The proposal distribution for MALA requires computation of the gradient of the log of the target density. The addition of a "drift" term that is a scalar multiple of this gradient directs the chain toward regions of higher π-probability mass. Specifically,

$$Q(\mathbf{x}, \cdot) \sim N(\mathbf{x} + \frac{\sigma_m}{2}\nabla \log(\pi(\mathbf{x})), \sigma_m \mathbf{\Sigma}_0). \quad (2)$$

There is a substantial body of work, both theoretical and empirical, on how to choose σ_m for RWM and MALA. The common guiding principle in these results is that an optimal MCMC sampler is one that generates output that is minimally autocorrelated. The reason for this view of optimality is that the statistical efficiency of estimates derived from MCMC samples increases with decreased autocorrelation.

Most of this work on scaling of MH proposals concerns the special case of $\mathbf{\Sigma}_0 = \mathbf{I}$, and most of the results are asymptotic as $d \to \infty$. In addition, analytical treatments generally only consider uncorrelated normal target distributions because of their mathematical simplicity relative to more complicated target distributions. We now briefly review some of these results.

2.1 Scaling Results for RWM

Several early studies have recommended rules of thumb for scaling RWM samplers (see Besag et al. 1995, for example). Roberts et al. (1997); Gelman et al. (1996) show analytically that the efficiency of RWM algorithms can be written as a function of their acceptance rates. They then derive the heuristic that σ_m should be set such that the sampler has an average acceptance rate of .234. This result applies to high-dimensional target distributions whose components are approximately iid, and the figure .234 is computed assuming the target is normal, but seems to hold fairly well in more practical settings.

When the components of π are approximately uncorrelated but heterogeneously scaled, Roberts and Rosenthal (2001) show that, for large d, the optimal RWM sampler still accepts at a rate of .234. However, they also show that when $\mathbf{\Sigma}_0 = \mathbf{I}$, RWM becomes less efficient as a function of the variability in the scales of the components of π. If, on the other hand, $\mathbf{\Sigma}_0$ is chosen to match the scaling of π, this inefficiency disappears.

This is an important observation for the practitioner. When the target distribution is the joint posterior distribution of parameters given a set of observations, there is usually no reason to think that the marginal posterior distributions will be of similar scale. Furthermore, it may be the case that there exist correlations in the posterior, in which case it is well-known that RWM can perform very poorly (Hills and Smith 1992; Roberts and Sahu 1997). There is therefore good reason to think that using $\mathbf{\Sigma}_0 = \mathbf{I}$ is a poor choice. It is this intuition in part that guides the development of the automatic tuning algorithm described in Section 3.

2.2 Scaling Results for Langevin Samplers

Like RWM, MALA samplers can in some sense be optimally scaled by making them accept at a theoretically-derived rate. For homogeneously-scaled iid target densities, as $d \to \infty$, the optimal acceptance rate for MALA samplers is .574 (Roberts and Rosenthal 1998). The considerably higher optimal acceptance rate (and thus higher optimal proposal variance) of MALA relative to RWM is of note because it helps explain why MALA mixes much faster than RWM.

The behavior of MALA, however, can at times be frustrating for the practitioner. One undesirable property of MALA samplers is that they are considerably more sensitive to scale heterogeneity than RWM (Roberts and Rosenthal 2001). This makes is all the more critical to find a good proposal covariance when using MALA.

A second property of MALA samplers that is a bit more subtle is that, unlike RWM, their characteristics in transience are vastly different than in stationarity. Whereas Roberts and Rosenthal (1998) show that optimal scaling in the stationary phase is given by $\sigma_m = \mathcal{O}(d^{-1/3})$, Christensen et al. (2005) show that the optimal scaling in the transient phase is $\sigma_m = \mathcal{O}(d^{-1/2})$. This seemingly benign difference has important consequences.

In almost all practical situations, MCMC samplers are initiated in a transient phase. This means that manual tuning via pilot runs, even if successful at achieving the desired acceptance rate in transience, will produce sub-optimal tuning for the stationary phase. Adding to this complication is the tendency for MALA to reject many consecutive proposals while in transience, producing chains that go for long periods without jumping at all. This property is sometimes referred to as "stickiness". See the left panel of Figure 2 for a typical example.

Stickiness makes it extremely difficult to tune MALA samplers because when scaled to perform well in stationarity, they simply will not jump for long periods after initialization. A simple solution proposed by Christensen et al. (2005) is to use $\sigma_m = \mathcal{O}(d^{-1/2})$ initially, and then switch to $\sigma_m = \mathcal{O}(d^{-1/3})$ once the chain has reached its stationary phase. The first obvious problem with this strategy is that is completely ignores the constants embedded in the notation.

The second, and more crippling, problem is that it is notoriously difficult to detect when a chain has reached stationarity. In practice, once one has obtained enough samples to make this determination, one probably already has enough samples from π to make inferences, so adjustment of σ_m is already moot.

An adaptive strategy for generating good proposals, however, should be able to scale the MALA algorithm such that it does not stick initially, and automatically adjust such that it accepts at an optimal rate throughout the run, obviating the need to assess convergence.

3 Automatic Tuning

There have been two general strategies for adaptive tuning of MH proposals that have been popular in the recent literature. The first is to run several parallel chains to form a replicated state space, and then use information from the ensemble of chains at their current states to construct a good proposal (Gilks et al. 1994; Gilks and Roberts 1996; Warnes 2001; Ter Braak 2006, for example). One advantage of this approach is that is preserves the Markov property of the sampler, so convergence to the desired target distribution is not an issue. The obvious disadvantage is that it requires computational effort that is multiplicative in the number of chains. Moreover, the number of chains required to produce effective samplers increases with the dimension of the parameter space. In particular, if the parameter space is high-dimensional, running many chains, each of which requires expensive likelihood computations, may not be feasible.

Because of the high computational cost of running parallel chains, we focus here on the second popular adaptive strategy of running a single chain that, at each iteration, uses previous states to generate a proposal (Haario et al. 2001; Atchadé and Rosenthal 2005; Atchadé 2006, for example). The complication is that this approach destroys the Markov property, so care is needed to make sure that the resultant processes are ergodic and hence converge to the desired target distributions. A simple solution is to simply stop adapting the proposal after a specified number of MC iterations (Gelfand and Sahu 1994), but this is somewhat un-satisfying as it requires a great deal of user intervention in determining when to stop the adaptation. Gilks et al. (1998) use the idea of Markov chain regeneration to show that the proposal can be altered infinitely often while still maintaining ergodicity, as long as the alterations happen at regeneration times. This strategy is of limited use, however, because regeneration times are extremely difficult to identify (Brockwell and Kadane 2005), and regenerations are too rare in high dimensions to be of practical use.

Another way of adapting proposals for single chains while preserving ergodicity is to use "controlled MCMC" (Andrieu and Robert 2001; Borkar 1991) with vanishing adaptation or "diminishing adaptation" (Roberts and Rosenthal 2008). The idea here is to attenuate the adaptation process such that the proposal distribution becomes approximately constant for large t. It is possible to show that many algorithms of this type are indeed ergodic (see Andrieu and Thoms 2008, and references therein). The most common mechanism for implementing vanishing adaptation is through Robbins-Monro recursion, borrowed from stochastic approximation (Benveniste et al. 1990). The general form of this recursion is

$$\theta_{(t+1)} \leftarrow \theta_{(t)} + \gamma_{(t)}(h(\theta_{(t)}) - \alpha), \tag{3}$$

where $h(\theta)$ is some some approximation to an unobservable function of interest $g(\theta)$ (with $\mathrm{E}[h(\theta)|\theta] = g(\theta)$), and it is used to find roots of the equation $g(\theta) - \alpha = 0$. Here, $\{\gamma_t,\ t = 1, 2, \ldots\}$ is a decreasing deterministic sequence of positive step sizes satisfying $\sum_{t=1}^{\infty} \gamma_{(t)} = \infty$ and $\sum_{t=1}^{\infty} \gamma_{(t)}^2 < \infty$.

Given the discussion in Section 2, a sensible way to proceed would be, first, to scale the MH proposal by letting $\theta = \sigma_m$, $h(\cdot)$ be an estimate of the sampler's acceptance rate and α be the optimal acceptance rate (.234 for RWM and .574

for MALA).

Next, a sensible way to adjust the shape of the proposal would be to let $\theta = h(\cdot) = \mathbf{\Sigma}_0$, and α be the covariance matrix that best approximates the shape of π. Since of course we do not know what this matrix is, we can estimate it using the sample covaraince matrix of the chain up until time (t).

This is in fact exactly the algorithm in Atchadé (2006) up to a few technical details. We modify this approach somewhat by using the log of σ_m for adaptation rather than σ_m itself, giving what we call Log Adaptive Proposals (LAP), Algorithm 2. We note that this algorithm is essentially the same as Algorithm 4 of Andrieu and Thoms (2008). The same LAP procedure may be used for MALA as well, although we recommend the slight alteration of initializing $\sigma^2_{m_{(0)}} = 2.4^2 d^{1/3}$ for MALA rather than $2.4^2/d$ for RWM. In addition, r_{opt} should be set to .234 for RWM, and .574 for MALA.

Algorithm 2 RWM with LAP algorithm

```
Require: Set initial values $\mathbf{X}_{(0)}$, $\sigma^2_{m_{(0)}} = \frac{2.4^2}{d}$, and $\mathbf{\Sigma}_{0_{(0)}} = \mathbf{I}_d$
1: for $t = 0$ to $T$ do
2:     take $k$ RWM steps using $\sigma^2_{m_{(t)}}$ and $\mathbf{\Sigma}_{0_{(t)}}$
3:     calculate $\hat{r}_{(t)} = \frac{\#\text{ jumps}}{k}$
4:     calculate $\hat{\mathbf{\Sigma}}_{0_{(t)}} = \frac{1}{k-1}(\mathbf{X}_{d\times k} - \bar{\mathbf{X}}_{(t)})(\mathbf{X}_{d\times k} - \bar{\mathbf{X}}_{(t)})^{\mathrm{T}}$
5:     calculate $\gamma_{1_{(t)}} = \frac{1}{t^{c_1}}$ and $\gamma_{2_{(t)}} = c_0\gamma_{1_{(t)}}$
6:     set $\log\sigma^2_{m_{(t+1)}} \leftarrow \log\sigma^2_{m_{(t)}} + \gamma_{2(t)}(\hat{r}_{(t)} - r_{\text{opt}})$
7:     $\mathbf{\Sigma}_{0_{(t+1)}} \leftarrow \mathbf{\Sigma}_{0_{(t)}} + \gamma_{1(t)}(\hat{\mathbf{\Sigma}}_{0_{(t)}} - \mathbf{\Sigma}_{0_{(t)}})$
8: end for
```

The intuition behind Algorithm 2 is quite simple. It takes a block of RWM steps, then estimates the acceptance rate for that block. If it accepts too often, it increases σ_m; if it accepts too rarely, it decreases σ_m.

It then computes the sample covariance matrix for that block of samples, and makes the the proposal covariance matrix $\mathbf{\Sigma}_0$ look a bit more like that sample covariance matrix. The idea here is that a good estimate of the covariance matrix that approximates the shape of the target π is the covariance matrix that is the best fit to the sample so far. We also note here that, in our experience, perhaps surprisingly, this algorithm works very well for target densities that depart severely from normality, including multi-modal densities.

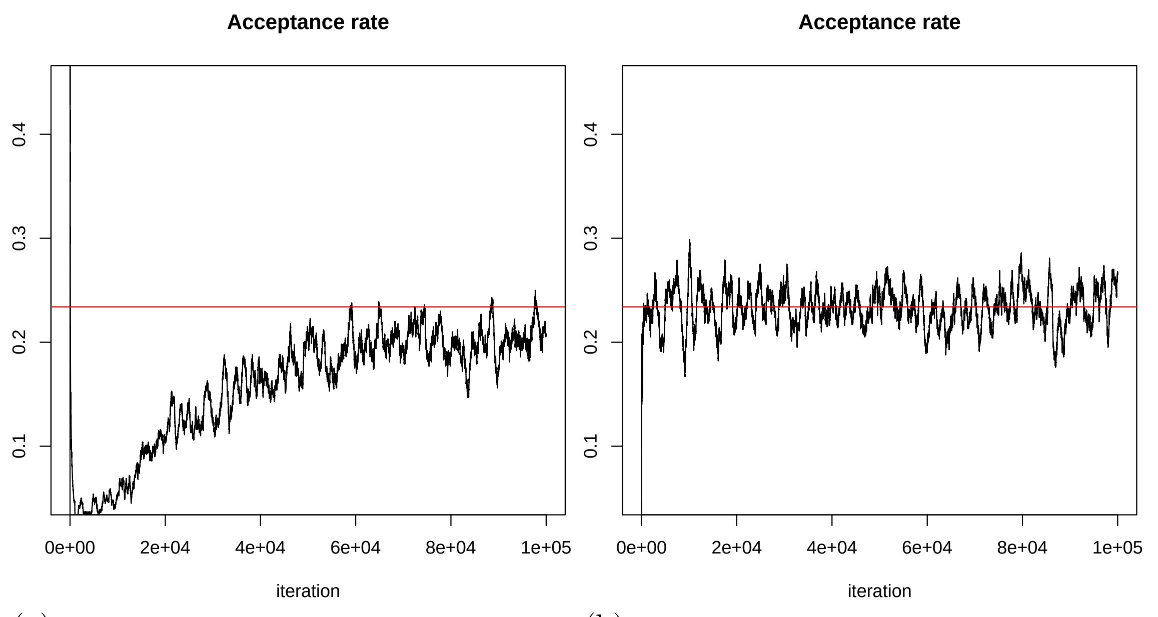
The only aspect of the LAP algorithm that is not completely automatic is the choice of the attenuation parameters $c_0 > 0$ and $c_1 \in (0, 1]$. In practice, we have found that these choices matter very little. We typically choose $c_0 = 1$ and $c_1 = .8$, and let it run without another thought.

Adapting the log of σ_m rather than σ_m itself has a huge effect (see Figure 1), and accomplishes two tasks. Firstly, it elegantly ensures that σ_m always remains positive. Secondly, and more importantly, it allows σ_m to make adjustments that are *multiplicative* rather than additive (this can be seen by exponentiating step 6 of Algorithm 2). We have found that for many target distributions arising from Bayesian data analysis, and in particular for MALA samplers, it is very difficult to determine *a priori* an appropriate starting value for σ_m, even to the nearest order of magnitude. The multiplicative adjustments allow the adaptive procedure to move quickly through orders of magnitude, and then quickly zero in on a good value by making large adjustments, in an absolute sense, when the optimal σ_m is large, and making small adjustments, in an absolute sense, when the optimal σ_m is small. Additive adjustments of the kind in Atchadé (2006) do not make these type of percentage-wise corrections.

Figure 1 demonstrates the effect of adapting σ_m on the log scale. When adapting σ_m directly (panel (a)), the process takes quite a long time to attain the optimal acceptance rate. In contrast, adapting the log (panel (b)) allows the acceptance rate to shoot quickly to optimality.

The effect of adaptation is quite striking for MALA samplers. In Figure 2 we plot the norm of vectors sampled, using MALA, from a 1000-dimensional standard normal distribution.

In panel (a), the MALA sampler was scaled with its "optimal" value in stationarity; that is, the value of σ_m for which the acceptance rate is .574 in the stationary phase of the chain. This plot shows the characteristic "stickiness"



(a) (b)

Figure 1: Acceptance rate as a function of iteration, moving average with a window width of 1000. The target is a mixture of normals. Panel (a) is from a RWM tuned by adapting σ_m directly, and (b) was tuned on the log scale with LAP. The horizontal line shows the "optimal" rate of .234

of MALA samplers, with the chain failing to jump even once until around iteration 20,000. This example, while dramatic, is not atypical. In panel (b), the MALA sampler is tuned with LAP. It is able to adjust its tuning quickly so that the sampler accepts jumps and hits stationarity almost immediately. It then automatically adjusts such that the optimal acceptance rate is achieved in the stationary phase. Thus, MALA tuned with LAP has the best of both worlds — fast convergence to stationarity, and optimal performace once therein.

4 Comparisons with other MCMC techniques

Now that we have demonstrated the efficacy of LAP for tuning MH proposals, we investigate how LAP compares with other methods for sampling from a posterior density. Specifically, we test RWM algorithms equipped with LAP (Algorithm 2) against slice sampling (Agarwal and Gelfand 2005) for geosta-

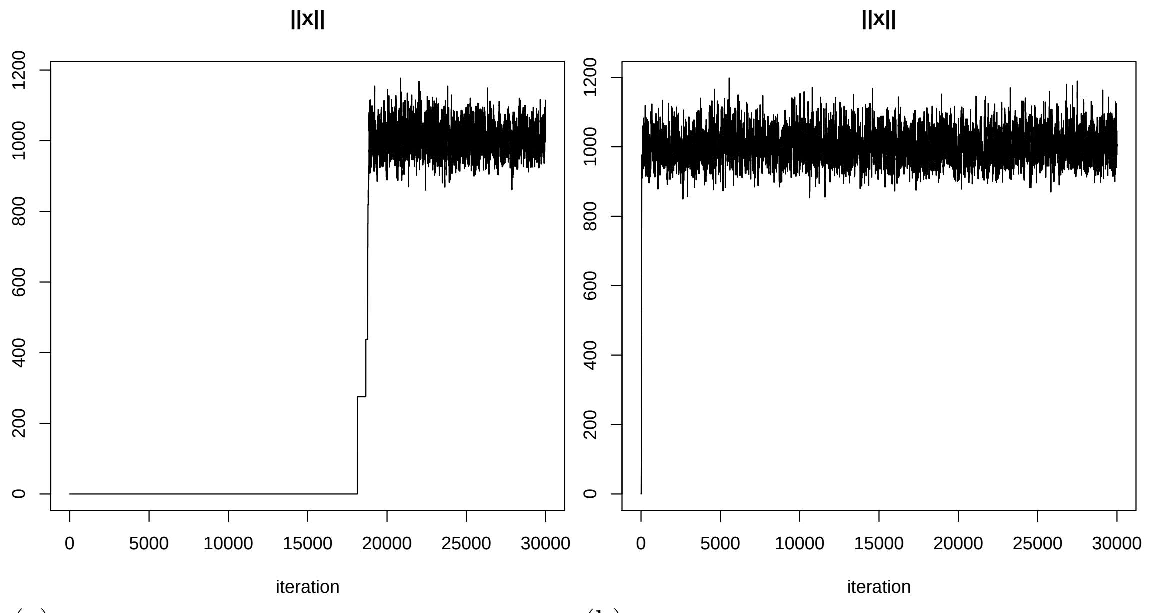


Figure 2: Samples from a standard normal distribution, $d = 1000$, using MALA. The chain in (a) is tuned "optimally" for the stationary phase, and (b) is automatically tuned with LAP.

tistical data, data augmentation (DA) (Albert and Chib 1993; Liu and Wu 1999) for binary response data, and adaptive rejection sampling (ARS) (Gilks and Wild 1992) for log-concave densities within a Gibbs sampler. We made an attempt to fit the same datasets analyzed in the original papers. Unfortunately, this was only possible in the case of the slice sampler because the other two datasets are no longer available. For the data augmentation case, we fit the dataset analyzed by Roy and Hobert (2007), who proved that parameter expansion data augmentation is a strict improvement over data augmentation. Finally, for the comparison with Gibbs sampling using adaptive rejection sampling, we use a popular example that is included with the `WinBUGS` (Lunn et al. 2000) software, to which `WinBUGS` applies its implementation of ARS. Thus, we compare popular methods to Metropolis sampling with LAP to "beat them at their own game."

These three somewhat specialized MCMC techniques were chosen for their popularity and good performance. Adaptive rejection sampling is undoubtedly the most often used, as it is the engine behind much of the popular `WinBUGS` (Lunn et al. 2000) software. The immense number of citations of Albert and Chib (1993) attests the popularity of data augmentation. Finally, slice sampling is the method of choice for sampling from densities arising from point-referenced spatial data in the recent book Banerjee et al. (2004).

Our basis for comparison among the different algorithms will essentially be how quickly they work. There are three factors governing speed. The first is simply how long it takes to compute each iteration of the algorithm. The second is how many iterations the sampler requires before it converges to stationarity. The third factor is how autocorrelated the sample is that the algorithm generates. The more autocorrelated the sample, the less information it contains, so we use the concept of effective sample size (Gelman and Rubin 1992; Chib and Carlin 1999; Sargent et al. 2000) to determine how informative a given sample is. Effective sample size (ESS) is defined as the actual size of the sample, divided by $1 + 2\sum_{k=1}^{\infty} \rho(k)$, where $\rho(k)$ is the autocorrelation at lag k. In practice, we compute this quantity by estimating the spectral density at 0 of an autoregressive model fitted to the chain, as implemented in the `R` package `coda` (R Development Core Team 2008; Plummer et al. 2007).

In each example, we conduct multiple pilot runs of the samplers to determine the burn-in period. We initialize each chain with starting parameter values that are dispersed widely throughout their support. We then calculate Gelman and Rubin shrink factors, sometimes referred to as $\hat{R}$ (Gelman and Rubin 1992). For each parameter we sample, we note the iteration at which the 97.5 percentile of the shrinkage factor drops below 1.2 (a standard rule of thumb), and choose the maximum over all the parameters. This value, plus a small safety margin, is chosen as the burn-in period.

Next, we run each algorithm to generate 50,000 post burn-in iterations (we found that the 5,000 iterations in Agarwal and Gelfand (2005) produced ESS estimates that were too variable). We use this sample to estimate ESS and

ESS per second. Finally, we use the burn-in period and the ESS per second estimates to calculate the time needed to produce an effective sample of size 500, which we call t500.

4.1 Data augmentation

Data augmentation for sampling from probit regression models was introduced by Albert and Chib (1993). Since then, it has been widely adopted as a method of choice when fitting models to binary data. The central idea is to add a latent variable to the model and condition on this new variable so that only draws from standard distributions are needed.

The model we fit is a probit regression model. Let $Y_1, \dots, Y_n$ be independent Bernoulli random variables, with $\Pr(Y_i = 1) = \Phi(x_i'\beta)$. As usual, $\Phi(\cdot)$ is the standard normal CDF, and x_i is the p-vector of covariates corresponding to Y_i. The parameter of interest is the p-vector β of regression coefficients. The likelihood is then

$$L(\beta; \mathbf{y}) = \prod_{i=1}^{n} \Phi(x_i'\beta)^{y_i}(1 - \Phi(x_i'\beta))^{1-y_i} \tag{4}$$

for $y_1, \dots, y_n \in \{0, 1\}$. We assign flat priors $\pi(\beta_j) \propto 1$ for $j = 1, \dots, p$.

Albert and Chib (1993) proposed a data augmentation algorithm (henceforth referred to here as DA) to sample from the posterior distribution $\pi(\beta|\mathbf{y})$. Denote by $TN(\mu, \sigma^2, w)$ the normal distribution with mean μ and variance σ^2, truncated to be positive if $w = 1$ and negative if $w = 0$. Also, let $\mathbf{X}$ be the design matrix with i^{th} row x_i'.

Algorithm 3 Data augmentation

```
Require: Compute $\hat{\mathbf{A}} = (\mathbf{X}'\mathbf{X})^{-1}$ and $\hat{\mathbf{B}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$.
1: for $i = 1$ to $N$ do
2:     Draw $z_1, \dots, z_n$ independently from $z_i \sim TN(x_i'\beta, 1, y_i)$.
3:     Draw $\beta$ from $N(\hat{\mathbf{B}}\mathbf{z}, \hat{\mathbf{A}})$
4: end for
```

Liu and Wu (1999) modified the DA algorithm by introducing another auxiliary variable. Their "parameter expansion" data augmentation (PX-DA) adds an additional draw from a gamma distribution, and it also simulates a Markov chain whose stationary distribution is the posterior. Let $G(a, b)$ be the gamma distribution with shape parameter a and rate parameter b.

Algorithm 4 Parameter expansion data augmentation

```
Require: Compute $\hat{\mathbf{A}} = (\mathbf{X}'\mathbf{X})^{-1}$ and $\hat{\mathbf{B}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$.
1: for $i = 1$ to $N$ do
2:     Draw $z_1, \ldots, z_n$ independently from $z_i \sim TN(x_i'\beta, 1, y_i)$.
3:     Draw $g^2$ from $G(\frac{n}{2}, \frac{1}{2}\sum_{i=1}^{n}[z_i - x_i'\hat{\mathbf{B}}\mathbf{z}]^2)$ and set $\mathbf{z}^* = g\mathbf{z}$
4:     Draw $\beta$ from $N(\hat{\mathbf{B}}\mathbf{z}^*, \hat{\mathbf{A}})$
5: end for
```

Roy and Hobert (2007) use asymptotic arguments to show that PX-DA should generate less autocorrelated samples than DA, a result that has been noted empirically (Liu and Wu 1999, e.g). Here, we compare both DA and PX-DA to a Metropolis sampler with LAP on the same lupus data from van Dyk and Meng (2001) that Roy and Hobert (2007) used.

The lupus data consists of $n = 55$ samples where $y_i, i = 1, \ldots, n$, is an indicator for latent membranous lupus nephritis, and x_{i1} and x_{i2}, $i = 1, \ldots, n$, are measured antibody levels. The unknown parameter is $\beta = (\beta_0, \beta_1, \beta_2)'$, where β_0 is the intercept.

Since the generated samples were so autocorrelated, we use 500,000 post burn-in iterations to compute effective sample size. Gelman-Rubin plots drop below 1.2 at 6,000 iterations for DA, 800 for PX-DA, and 300 for Metropolis. Based on this criterion, we select burn-in periods of 7,000 for DA and 1,000 for PX-DA. Because the adaptation of the Metropolis proposal did not stabilize until about 800 iterations, we extended the burn-in for the Metropolis sampler to 1,000 iterations.

Results of timing experiments are shown in Tables 1 and 2. We find that PX-DA is dramatically faster than DA, though not as dramatically as Roy and

Hobert (2007) report. Metropolis with LAP, on the other hand, is staggeringly more efficient than the data augmentation algorithms.

Table 1: Comparing RWM with LAP to DA and PX-DA in terms of ESS/sec.

	β_0	β_1	β_2
DA	0.46	0.28	0.47
PX-DA	16.16	15.72	16.08
RWM	525.98	626.44	555.71

Table 2: Comparing RWM with LAP to DA and PX-DA in terms of time required to generate 500 effective samples.

	β_0	β_1	β_2
DA	1089.15	1772.27	1063.26
PX-DA	32.30	33.15	32.44
RWM	1.09	0.94	1.04

Metropolis is both computationally much faster per iteration and shows much less autocorrelation than DA and PX-DA. The difference in computational time is surprising, given the simplicity of DA and PX-DA. Further investigation revealed that well over 90% of compute time for DA and PX-DA goes into drawing from the truncated normal distributions. We implemented these draws with the function `rtnorm` from the `R` package `msm` (R Development Core Team 2008; Jackson 2008). It is possible that `rtnorm` is a slow implementation. However, even if we assumed that drawing from the truncated normal took no time whatsoever, it would not be enough to make PX-DA equal the efficiency of the Metropolis sampler.

Although DA can be extended to a small class of other models (Albert and Chib 1993), it has nowhere near the generality of Metropolis. For binary data, for example, Metropolis permits the use of a logit link instead of the probit. In addition, one might wish to use something other than uniform priors on β, which is easily accomodated by Metropolis samplers. Simply, dramatically better performance and greater flexibility make the Metropolis algorithm with LAP greatly preferable to data augmentation algorithms for Bayesian fitting of binary data.

4.2 Slice sampling

In this example, we model data as a realization of a Gaussian random field. The observations are assumed to be jointly normal, with a constant mean and a covariance structure that depends on the distances between observations.

Let $Y_1, \ldots, Y_n$ be a sample from a Gaussian random field with a constant mean, and let $\mathbf{s}_1, \ldots, \mathbf{s}_n$ be the set of corresponding locations. Then

$$\mathbf{Y} \sim N(\mu \mathbf{1}, \boldsymbol{\Sigma}(\boldsymbol{\theta})), \tag{5}$$

where $\boldsymbol{\Sigma}(\boldsymbol{\theta})_{ij} = C(\boldsymbol{\theta}; \mathbf{s}_i, \mathbf{s}_j)$ for some covariance function C. We let C be the popular Matèrn covariance function (see Stein 1999, e.g.) with a nugget,

$$C(\boldsymbol{\theta}; \mathbf{s}_i, \mathbf{s}_j) = \left(\frac{(\|\mathbf{s}_i - \mathbf{s}_j\|/\phi)^\nu}{2^{\nu-1}\Gamma(\nu)} \right) \mathcal{K}_\nu \left(\frac{\|\mathbf{s}_i - \mathbf{s}_j\|}{\phi} \right) + \tau^2 \mathbb{1}_{\mathbf{s}_i = \mathbf{s}_j}, \tag{6}$$

where $\mathcal{K}_\nu$ is the modified Bessel function of the second kind of order ν, and $\boldsymbol{\theta} = (\sigma^2, \phi, \nu, \tau^2)'$. ϕ is called the range parameter, and it determines how quickly the correlation falls off as a function of distance. ν governs the smoothness of the process, and τ^2 is called the nugget and could represent measurement error.

An important special case of (6) is when ν is fixed at .5, called the exponential covariance function. We are interested here in both the full Matèrn and the exponential covariance functions.

Agarwal and Gelfand (2005) propose to use slice sampling for computations on this model. Slice sampling is an auxiliary variable method that works by "knocking out" the likelihood. Consider the parameter vector $\boldsymbol{\theta}^* = (\boldsymbol{\theta}\, \mu)'$, and denote the likelihood $L(\boldsymbol{\theta}^*; \mathbf{Y})$ and the prior $\pi(\boldsymbol{\theta})$. The idea behind slice sampling is to draw from a single auxiliary variable U which, conditional on $\mathbf{Y}$ and $\boldsymbol{\theta}^*$, is distributed uniformly on $(0, L(\boldsymbol{\theta}^*; \mathbf{Y}))$ (see Agarwal and Gelfand (2005) for details). Agarwal and Gelfand (2005) further introduce "shrinkage sampling" as a way to constrain the support over which certain draws are made, increasing the probability of good draws.

Their version of slice sampling begins by partitioning $\boldsymbol{\theta}^*$ as $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$, such that $f(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2, \mathbf{Y})$ is a simple distribution from which to sample. We then execute Algorithm (5) until we have enough samples to estimate posterior quantities of interest.

Algorithm 5 Slice sampler with shrinkage sampling

```
Require: Starting values $\boldsymbol{\theta}^* = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$
1: for $i = 1$ to $N$ do
2:     Compute $V \leftarrow -\log L(\boldsymbol{\theta}^*; \mathbf{Y}) + z$, where $z \sim \text{Exp}(1)$.
3:     Draw $\boldsymbol{\theta}_2$ from the prior $\pi(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1)$, subject to the constraint that $-\log L(\boldsymbol{\theta}^*) <$
       $V < \infty$, using shrinkage sampling.
4:     Draw $\boldsymbol{\theta}_1$ from $f(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2, \mathbf{Y})$.
5: end for
```

The exponential random variable in step (2) arises, in place of a uniform random variable, because we prefer to evaluate the log likelihood instead of the likelihood itself.

We note that step (4) in the slice sampling algorithm only works well if the full conditional for $\boldsymbol{\theta}_1$ is a standard distribution. In particular, that restricts us to the use of a conjugate prior. If this full conditional is not a standard distribution, one must still draw from it, possibly using one or several Metropolis steps.

We apply slice sampling and Metropolis with LAP to fit the model defined by (5) and (6) to the scallop data of Ecker and Heltshe (1994). We fit this data twice, once keeping ν is fixed at .5, and once assuming ν is unknown.

For the slice sampler, we determined an appropriate burn-in time of 800 iterations for the exponential covariance model and 500 for the full Matèrn model. For the exponential model, this is considerably more than the 500 used by Agarwal and Gelfand (2005). The Metropolis sampler required many more iterations before it converged, 5,000 for the exponential model and 10,000 for the Matèrn model.

Agarwal and Gelfand (2005) present two different versions of the slice sampler with shrinkage sampling. One version updates each parameter separately, and the other updates σ^2 and τ^2 simultaneously. In our experiments, the former performed uniformly better than the latter, so it is the version with individual updates that we use for comparisons here.

In general, slice sampling produced samples that were much less autocorrelated than Metropolis with LAP. However, per iteration, Metropolis is many times faster that slice sampling. The sum of these two competing effects is that the performance of the two algorithms is roughly comparable.

Table 3: Comparing RWM with LAP to Slice sampling in terms of ESS/sec.

		σ^2	ϕ	τ^2	ν
Exponential	Slice	0.78	0.81	3.64	-
	RWM	2.27	2.01	6.10	-
Matèrn	Slice	0.77	0.57	1.72	0.68
	RWM	0.54	0.38	0.61	0.76

Table 4: Comparing RWM with LAP to Slice sampling in terms of time required to generate 500 effective samples.

		σ^2	ϕ	τ^2	ν
Exponential	Slice	710.37	687.28	204.95	-
	RWM	247.43	276.50	109.10	-
Matèrn	Slice	755.12	978.28	393.04	840.47
	RWM	1160.79	1564.33	1060.28	897.41

Results of the timing experiments are displayed in Tables 3 and 4. For the exponential model, Metropolis beats slice sampling handily in terms of both ESS/sec and t500. However, with the exception of the ν parameter, slice sampling is faster than Metropolis on the Matèrn model. This happens primarily because in the Matèrn case, computational effort at each iteration is dominated by the evaluation of the Bessel function (about 70% of the total compute time). For the slice sampler, which runs much more slowly per iteration to start with, the effect of evaluating the Bessel function has less of an impact.

Agarwal and Gelfand (2005) present slice sampling as an off-the-shelf algorithm that efficiently samples from models with unknown covariance parameters. This claim is true in that it requires almost no tuning, and it produces samples with low autocorrelations. However, Metropolis sampling with LAP seems to perform just as well, and also requires no tuning. Furthermore, slice sampling can only be used in limited settings, even within the context of spatial data. For example, for a generalized linear model with a random effect modeled as a Gaussian random field, slice sampling cannot be applied (Agarwal and Gelfand 2005). Metropolis sampling, however, has no such limitations.

In addition, if one wishes to avoid making Metropolis steps from within the slice sampler, one is required to use a conjugate prior for $\boldsymbol{\theta}_1$ in step (4) of Algorithm (5). In the example above, that means using inverse gamma priors, which are known to cause problems for variance components (see Gelman 2006, e.g.). Metropolis algorithms, of course, impose no such restrictions on priors.

4.3 Gibbs sampling with adaptive rejection sampling

Adaptive rejection sampling (Gilks and Wild 1992; Gilks 1992) is a popular tool for sampling from univariate log-concave densities. Its utility for Bayesian computations is apparent when it is placed within a Gibbs sampler for models where the full conditional distribution for one or more of the parameters is not easily obtained via conjugacy, but which is known to be log-concave. In fact, ARS is the algorithm employed by `WinBUGS` (Lunn et al. 2000) in these situations. Essentially, ARS is a traditional rejection sampler that reduces the frequency of rejection by building a picture of the target density using user-supplied derivative information.

Suppose we want to sample from a univariate distribution Π with unnormalized density $\pi(x)$. A rejection sampler uses an "envelope" function $\pi_u(x)$, with $\pi_u(x) \geq \pi(x)$, and a "squeezing" function $\pi_\ell(x)$, with $\pi_\ell(x) \leq \pi(x)$, for all x in the support of Π.

The rejection sampler independently samples a point x^* from $\pi_u(x)$ and y from a uniform $(0,1)$ distribution. Then, if $y \leq \pi_\ell(x^*)/\pi_u(x^*)$, we accept x^*. If not, we compute $\pi(x^*)$ and accept only if $y \leq \pi(x^*)/\pi_u(x^*)$. This algorithm is only useful if we can easily define and sample from the envelope function $\pi_u(x)$. The original version of adaptive rejection sampling (Gilks and Wild 1992) uses user-supplied derivatives of $\pi(x)$ to construct piecewise-linear envelope and squeezing functions $\pi_u(x)$ and $\pi_\ell(x)$.

To compare ARS to RWM with MALA, we analyze the famous stackloss data from Brownlee (1960). This dataset consists of 21 measurements of "stackloss" and three covariates: air flow, temperature, and acid concentration. The model we apply is taken directly from the "Stacks" example provided with `WinBUGS`. Due to the suspected presence of outliers in the data vector $\mathbf{y} = y_1, \ldots, y_n$, we model model the response as having a Laplace distribution. Letting $\mathbf{X}$ be the $n \times 4$ matrix of centered and scaled covariates, with the leftmost column consisting of ones,

$$\begin{aligned}
y_i &\sim \text{Laplace}(\mu_i, s) \quad \text{independently for } i = 1, \ldots, 21 \\
\mu_i &= \sum_{j=0}^{3} \beta_j x_{ij} \\
\beta_j &\sim N(0, \tau_0) \\
s &\sim \text{Exponential}(\lambda_0), \qquad (7)
\end{aligned}$$

where τ_0 is the precision of the normal prior, and λ_0 is the rate of the Exponential prior.

We deviate slightly from the Stacks example in that we replace their Gamma prior on s with an Exponential prior because the Gamma density is not log-concave for $s < 1$, which would preclude the use of ARS. We choose $\tau_0 = 0.00001$ as in the Stacks example, and $\lambda_0 = 0.01$, which gives very similar parameter estimates to the model with the Gamma prior.

The implementation we use for ARS is from the `R` package `ars` (Rodriguez 2009). The required burn-in period for the Gibbs sampler with ARS was extremely short, about 300 iterations. In contrast, RWM with LAP required 5,000 iterations based on Gelman and Rubin shrink factors.

Table 5: Comparing RWM with LAP to ARS in terms of ESS/sec.

	β_0	β_1	β_2	β_3	s
GS/ARS	85.51	29.08	31.85	72.68	100.93
RWM	1015.33	122.73	117.78	133.34	117.48

Table 6: Comparing RWM with LAP to ARS in terms of time required to generate 500 effective samples.

	β_0	β_1	β_2	β_3	s
GS/ARS	7.87	19.22	17.72	8.91	6.98
RWM	0.99	4.58	4.75	4.25	4.76

The Gibbs sampler with ARS did a phenomenal job of generating posterior samples with low autocorrelations. However, each iteration of the Gibbs sampler took much longer to compute than each iteration of RWM. This is true for several reasons. First, at each iteration, the Gibbs sampler has to make a draws from all five parameters in the model defined by (7), instead of just a single draw for each iteration of RWM. In the best case, this requires five times more likelihood evaluations per iteration. Second, each ARS draw within a single Gibbs iteration requires several likelihood and derivative calculations to construct the envelope and squeezing functions, in addition to the computation needed for the sampling step in ARS. Finally, at each Gibbs iteration, ARS may reject a few samples before it accepts one, further increasing computing time per iteration.

Overall, Tables 5 and 6 show that RWM with LAP was considerably more efficient for all parameters in model defined by (7), in terms of both ESS/sec and t500. It is possible that a portion of the clear discrepancies in computational efficiency is due to the overhead involved in the `R` interface with the compiled ARS code. However, it seems that for this model, the Metropolis sampler with LAP handily out-performs the Gibbs sampler with ARS.

In addition to the its computational advantages over the Gibbs sampler with ARS, RWM is again considerably more flexible in the kinds of models it permits. As the example highlights, the commonly-used Gamma prior will not work with ARS (of course, the Gamma prior is most often used as half of a conjugate pair whose full conditional distribution can be sampled directly). We also note that `WinBUGS` uses an implementation of ARS that makes several draws at each Gibbs iteration, seeking to accept samples that are negatively autocorrelated with the existing chain. This strategy takes longer per iteration, but may justify its expense by generating a sample that has extremely low autocorrelations. We have made no attempt to implement this strategy here. Finally, we note that more flexible versions of ARS do exist, which do not require user-supplied derivative functions (Gilks 1992) (this is actually the version implemented in `WinBUGS`) and which do not require log-concavity (Gilks et al. 1995), although the latter algorithm is not as commonly used.

5 Discussion

The Metropolis Hastings algorithm is one of the oldest and simplest MCMC techniques available. When equipped with a mechanism like LAP for generating good proposals, MH may also be the best-performing MCMC technique available. We have reviewed some important theoretical work concerning the efficiency of MH algorithms, and used some of these results to recommend LAP, an adaptive technique to optimize proposal distributions. We have argued that tuning RWM and MALA is a crucial, though perhaps under-appreciated, exercise, and that, with LAP, it does not have to be a painful one. Finally, whereas others have shown simply (and importantly) that adaptive MCMC algorithms outperform their non-adaptive analogues (Haario et al. 2001; Atchadé 2006; Andrieu and Thoms 2008; Roberts and Rosenthal 2008, e.g.), we have compared the performance of RWM with LAP to three commonly-used non-Metropolis high-performance MCMC algorithms and concluded that RWM with LAP is preferable both for its computational efficienty and its superior flexibility.

References

Agarwal, D. K. and Gelfand, A. E. (2005), "Slice sampling for simulation based fitting of spatial data models," *Stat. Comput.*, 15, 61–69.

Albert, J. H. and Chib, S. (1993), "Bayesian analysis of binary and polychotomous response data," *J. Amer. Statist. Assoc.*, 88, 669–679.

Andrieu, C. and Robert, C. P. (2001), "Controlled MCMC for optimal scaling," Tech. Rep. 0125, Iniversité Paris Dauphine, Ceremade.

Andrieu, C. and Thoms, J. (2008), "A tutorial on adaptive MCMC," *Stat. Comput.*, 18, 343–373.

Atchadé, Y. F. (2006), "An adaptive version for the metropolis adjusted Langevin algorithm with a truncated drift," *Methodol. Comput. Appl. Probab.*, 8, 235–254.

Atchadé, Y. F. and Rosenthal, J. S. (2005), "On adaptive Markov chain Monte Carlo algorithms," *Bernoulli*, 11, 815–828.

Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2004), *Hierarchical Modeling and Analysis for Spatial Data*, Monographs on Statistics and Applied Probability, Boca Raton: Chapman & Hall/CRC.

Banerjee, S., Gelfand, A. E., Finley, A. O., and Sang, H. (2008), "Gaussian Predictive Process Models for Large Spatial Data Sets," *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 70, 825–848.

Benveniste, A., Métivier, M., and Priouret, P. (1990), *Adaptive algorithms and stochastic approximations*, vol. 22 of *Applications of Mathematics (New York)*, Berlin: Springer-Verlag, translated from the French by Stephen S. Wilson.

Besag, J. (1994), "Comments on "Representations of knowledge in complex systems"; by U. Grenander and M.I. Miller," *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 56, 591–592.

Besag, J., Green, P., Higdon, D., and Mengersen, K. (1995), "Bayesian computation and stochastic systems," *Statist. Sci.*, 10, 3–66, with comments and a reply by the authors.

Borkar, V. S. (1991), *Topics in controlled Markov chains*, vol. 240 of *Pitman Research Notes in Mathematics Series*, Harlow: Longman Scientific & Technical.

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

Brownlee, K. A. (1960), *Statistical theory and methodology in science and engineering*, A Wiley Publication in Applied Statistics, New York: John Wiley & Sons Inc.

Chib, S. and Carlin, B. P. (1999), "On MCMC sampling in hierarchical longitudinal models," *Statistics and Computing*, 9, 17–26.

Christensen, O. F., Roberts, G. O., and Rosenthal, J. S. (2005), "Scaling limits for the transient phase of local Metropolis-Hastings algorithms," *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67, 253–268.

Christensen, O. F. and Waagepetersen, R. (2002), "Bayesian prediction of spatial count data using generalized linear mixed models," *Biometrics*, 58, 280–286.

Ecker, M. D. and Heltshe, J. F. (1994), "Geostatistical estimates of scallop abundance," in *Case studies in biometry*, eds. Lange, N., Ryan, L., Billard, L., Brillinger, D., Conquest, L., and Greenhouse, J., New York: Wiley, pp. 107–124.

Gelfand, A. E. and Sahu, S. K. (1994), "On Markov chain Monte Carlo acceleration," *J. Comput. Graph. Statist.*, 3, 261–276.

Gelman, A. (2006), "Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper)," *Bayesian Anal.*, 1, 515–533 (electronic).

Gelman, A., Roberts, G. O., and Gilks, W. R. (1996), "Efficient Metropolis jumping rules," in *Bayesian statistics, 5 (Alicante, 1994)*, New York: Oxford Univ. Press, Oxford Sci. Publ., pp. 599–607.

Gelman, A. and Rubin, D. B. (1992), "Inference from iterative simulation using multiple sequences," *Stat. Sci.*, 7, 457–472.

Gilks, W. R. (1992), "Derivative-free Adaptive Rejection Sampling for Gibbs Sampling," in *Bayesian Statistics 4. Proceedings of the Fourth Valencia International Meeting*, eds. Bernardo, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M., Clarendon Press [Oxford University Press], pp. 641–649.

Gilks, W. R., Best, N., and Tan, K. (1995), "Adaptive rejection Metropolis sampling," *Appl. Stat.*, 44, 455–472.

Gilks, W. R. and Roberts, G. O. (1996), "Strategies for improving MCMC," in *Markov Chain Monte Carlo in Practice*, eds. Gilks, W. R., Richardson, S., and Spiegelhalter, D., London: Chapman Hall, pp. 89–114.

Gilks, W. R., Roberts, G. O., and George, E. I. (1994), "Adaptive Direction Sampling," *The Statistician*, 43, 179–189.

Gilks, W. R., Roberts, G. O., and Sahu, S. K. (1998), "Adaptive Markov chain Monte Carlo through regeneration," *J. Amer. Statist. Assoc.*, 93, 1045–1054.

Gilks, W. R. and Wild, P. (1992), "Adaptive rejection sampling for Gibbs sampling," *Appl. Statist.*, 41, 337–348.

Gottardo, R., Raftery, A. E., Yeung, K. Y., and Bumgarner, R. E. (2006), "Quality control and robust estimation for cDNA microarrays with replicates," *J. Amer. Statist. Assoc.*, 101, 30–40.

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

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

Hills, S. E. and Smith, A. F. M. (1992), "Parameterization issues in Bayesian inference," in *Bayesian Statistics, 4 (Peñíscola, 1991)*, New York: Oxford Univ. Press, pp. 227–246.

Jackson, C. (2008), *msm: Multi-state Markov and hidden Markov models in continuous time*, R package version 0.8.1.

Liu, J. S. and Wu, Y. N. (1999), "Parameter expansion for data augmentation," *J. Amer. Statist. Assoc.*, 94, 1264–1274.

Lunn, D. J., Thomas, A., Best, N., and Spiegelhalter, D. (2000), "WinBUGS – a Bayesian modelling framework: Concepts, structure, and extensibility," *Statistics and Computing*, 10, 325–337.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953), "Equation of State Calculations by Fast Computing Machines," *J. Chem. Phys.*, 21, 1087–1092.

Nott, D. J., Yu, Z., Chan, E., Cotsapas, C., Cowley, M. J., Pulvers, J., Williams, R., and Little, P. (2007), "Hierarchical Bayes variable selection and microarray experiments," *J. Multivariate Anal.*, 98, 852–872.

Plummer, M., Best, N., Cowles, K., and Vines, K. (2007), *coda: Output analysis and diagnostics for MCMC*, R package version 0.12-1.

R Development Core Team (2008), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Roberts, G. and Rosenthal, J. (2008), "Examples of Adaptive MCMC," *J. Comput. Graph. Statist.*, 18, 349–367.

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

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

— (2001), "Optimal scaling for various Metropolis-Hastings algorithms," *Statist. Sci.*, 16, 351–367.

Roberts, G. O. and Sahu, S. K. (1997), "Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler," *J. Roy. Statist. Soc. Ser. B*, 59, 291–317.

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

Rodriguez, P. P. (2009), *ars: Adaptive Rejection Sampling*, original C++ code from Arnost Komarek based on ars.f written by P. Wild and W. R. Gilks. R package version 0.4.

Roy, V. and Hobert, J. P. (2007), “Convergence rates and asymptotic standard errors for Markov chain Monte Carlo algorithms for Bayesian probit regression,” *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 69, 607–623.

Royle, J. A. and Wikle, C. K. (2005), “Efficient statistical mapping of avian count data,” *Environ. Ecol. Stat.*, 12, 225–243.

Sargent, D. J., Hodges, J. S., and Carlin, B. P. (2000), “Structured Markov chain Monte Carlo,” *J. Comput. Graph. Statist.*, 9, 217–234.

Stein, M. L. (1999), *Interpolation of spatial data*, Springer Series in Statistics, New York: Springer-Verlag, some theory for Kriging.

Ter Braak, C. J. F. (2006), “A Markov chain Monte Carlo version of the genetic algorithm differential evolution: easy Bayesian computing for real parameter spaces,” *Stat. Comput.*, 16, 239–249.

van Dyk, D. A. and Meng, X.-L. (2001), “The art of data augmentation,” *J. Comput. Graph. Statist.*, 10, 1–111, with discussions, and a rejoinder by the authors.

Warnes, G. R. (2001), “The Normal Kernel Coupler: An adaptive Markov Chain Monte Carlo method for eciently sampling from multi-modal distributions,” Tech. Rep. 39, University of Washington Department of Statistics.