Spiegelhalter, D. J., Thomas, A., Best, N. G. and Gilks, W. R. (1994) *BUGS: Bayesian inference Using Gibbs Sampling*. Cambridge: MRC Biostatistics Unit.

Tierney, L. (1991) Exploring posterior distributions using Markov chains. In *Computer Science and Statistics: Proc. 23rd Symp. Interface* (ed. E. Keramidas), pp. 563–570. Fairfax Station: Interface Foundation.

Wakefield, J. C., Gelfand, A. E. and Smith, A. F. M. (1991) Efficient generation of random variates via the ratio-of-uniforms method. *Statist. Comput.*, **1**, 129–133.

Zeger, S. and Karim, M. R. (1991) Generalised linear models with random effects: a Gibbs sampling approach. *J. Am. Statist. Ass.*, **86**, 79–86.

6

Strategies for improving MCMC

Walter R Gilks
Gareth O Roberts

6.1 Introduction

In many applications raw MCMC methods, in particular the Gibbs sampler, work surprisingly well. However, as models become more complex, it becomes increasingly likely that untuned methods will not *mix* rapidly. That is, the Markov chain will not move rapidly throughout the support of the target distribution. Consequently, unless the chain is run for very many iterations, Monte-Carlo standard errors in output sample averages will be large. See Roberts (1995) and Tierney (1995) in this volume for further discussion of Monte-Carlo standard errors and Markov chain mixing.

In almost any application of MCMC, many models must be explored and refined. Thus poor mixing can be severely inhibiting. Run times of the order of seconds or minutes are desirable, runs taking hours are tolerable, but longer run times are practically impossible to work with. As models become more ambitious, the practitioner must be prepared to experiment with strategies for improving mixing. Techniques for reducing the amount of computation per iteration are also important in reducing run times.

In this chapter, we review strategies for improving run times of MCMC. Our aim is to give sufficient detail for these strategies to be implemented: further information can be found in the original references. For readers who are new to MCMC methodology, we emphasize that familiarity with the material in this chapter is not a prerequisite for successful application of MCMC; Gilks *et al.* (1995b: this volume) provide enough information to permit application of MCMC in straightforward situations.

For simplicity, we will mostly assume that the Markov chain takes values in k-dimensional Euclidean space $\mathbb{R}^k$, although most of the techniques we discuss apply more generally. The target density (for example a posterior distribution) is denoted $\pi(.)$, and $X_t = (X_{t.1}, X_{t.2}, \ldots, X_{t.k})^T$ denotes the state of the chain at iteration t. Note that throughout this chapter $X_{t.i}$ will denote a scalar. A generic point in the sample space will be denoted $X = (X_{.1}, X_{.2}, \ldots, X_{.k})^T$.

6.2 Reparameterization

In this section, we consider simple techniques for reparameterizing commonly used models to improve mixing. We begin by examining the consequences of correlations among the $\{X_{.i}\}$.

6.2.1 Correlations and transformations

Both the Gibbs sampler and the Metropolis–Hastings algorithm may be sensitive to the choice of parameterization of the model. Consider for example the two-dimensional target density $\pi(.)$ illustrated in Figure 6.1(a). The ellipses represent contours of $\pi(.)$, and indicate that the probability in $\pi(.)$ is concentrated around the diagonal line $X_{.1} = X_{.2}$.

To sample from this target distribution, we might use a Gibbs sampler, updating one component of X at a time. Suppose that, at iteration t, the current point X_t is at the intersection of the arrows in Figure 6.1(a). Updating one of the components of X_t will produce a new point X_{t+1} lying in the direction of one of the arrows (depending on which component is updated). Since $\pi(.)$ is concentrated around the diagonal, the full conditional densities $\pi(X_{t+1.1}|X_{t.2})$ and $\pi(X_{t+1.2}|X_{t.1})$ will be concentrated near X_t (see Figure 6.1(c)) and so X_{t+1} will probably lie quite close to the current point. This will tend to happen at every iteration, so the Gibbs chain will move around rather slowly, in general taking small steps so as to stay close to the main diagonal. The consequence of this slow mixing is that a long simulation will be required to obtain adequate precision in the output analysis.

Alternatively, a Metropolis algorithm might be used for this problem. Suppose, for simplicity, that the Metropolis proposal comprises a uniform distribution on a disc centred on the current point X_t. This is indicated by the circle in Figure 6.1(a). As most of the disc lies away from the diagonal, the proposal distribution will tend to generate candidate points X' for which $\pi(X')$ is small compared to $\pi(X_t)$. Such candidates will probably be rejected (the acceptance probability being $\min[1, \pi(X')/\pi(X_t)]$) and then X_{t+1} will be identical to X_t. Thus the chain is likely to get stuck at X_t for several iterations. When a candiate point is eventually accepted, it will probably lie close to the main diagonal, and the same problem will recur at the next iteration. Consequently, this Metropolis algorithm will mix slowly.

A solution to the slow mixing of the Gibbs sampler and Metropolis algorithms in this example is to transform X to a new variable Y. Let

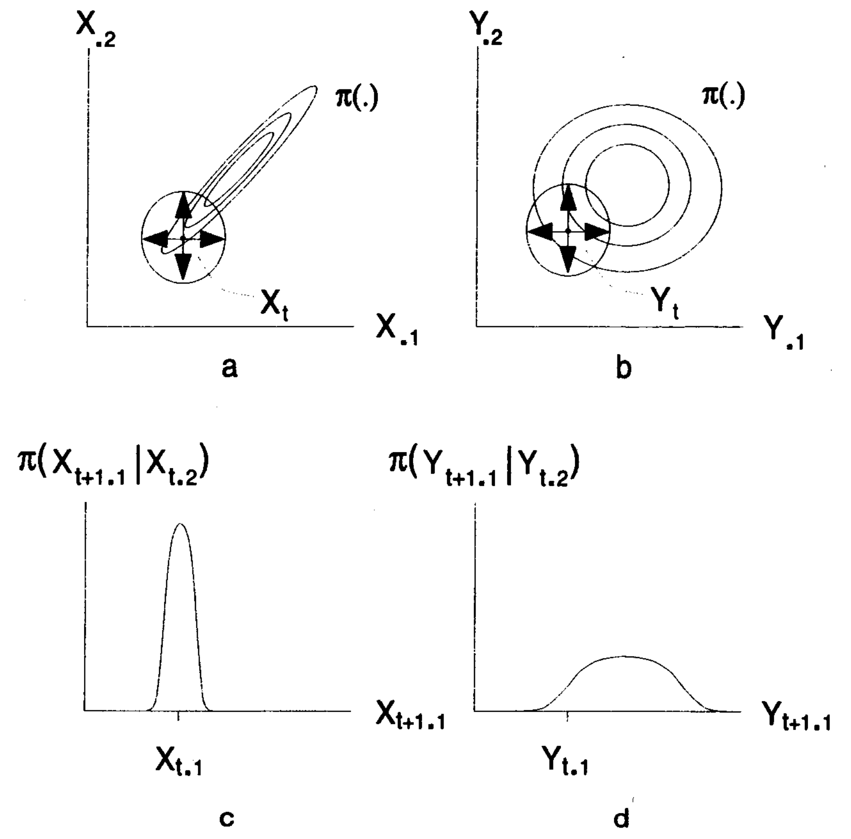

Figure 6.1 ***Illustrating Gibbs sampling and Metropolis algorithms for a bivariate target density $\pi(.)$. Contours of $\pi(.)$: (a) before reparameterization; (b) after reparameterization. Full conditional densities at time t: (c) before reparameterization; (d) after reparameterization. See text for explanation.***

$Y_{.1} = X_{.1} + X_{.2}$ and $Y_{.2} = 3[X_{.1} - X_{.2}]$. Contours for $\pi(Y)$ are approximately drawn in Figure 6.1(b). These contours are much more open, and indicate absence of correlation between the components of Y under $\pi(.)$. The Gibbs sampler will now be able to move around freely since moves away from the diagonal are no longer inhibited (as indicated by the full conditional density drawn in Figure 6.1(d)). Moreover, candidate points generated using the original Metropolis proposal density will be rejected less often.

For the Metropolis algorithm, an alternative strategy to improve mixing would be to use a different proposal distribution. For example, for the target distribution shown in Figure 6.1(a), a uniform proposal distribution on a diagonally oriented elliptical disc would generate a more rapidly mixing chain. However, using a transformed proposal for the original variable X is equivalent to using the original proposal for a transformed variable $Y(X)$. For present purposes, it is convenient to think in terms of variable transformation, since thereby we can address the issue of mixing in both Gibbs samplers and Metropolis–Hastings algorithms simultaneously.

After running the MCMC for the transformed variable Y, the original variables X can be recovered simply by transforming back the values in the MCMC output; (we assume that a unique inverse transform $X(Y)$ exists). In the above example, this would involve calculating for each iteration t: $X_{t.1} = \frac{1}{2}Y_{t.1} + \frac{1}{6}Y_{t.2}$ and $X_{t.2} = \frac{1}{2}Y_{t.1} - \frac{1}{6}Y_{t.2}$.

Placing the above discussion in a Bayesian context, variable X represents a vector of parameters; $\pi(X)$ is a posterior distribution; Figure 6.1(a) illustrates a strong posterior correlation; and any one-to-one transformation $Y(X)$ represents a *reparameterization*. Below, we discuss reparameterization strategies designed to reduce posterior correlations in commonly used models. However, posterior correlations are not the only cause of slow mixing; we return to this point in Section 6.2.5.

6.2.2 Linear regression models

Problems of slow mixing can occur in the simplest of statistical models. Consider for example the linear regression model

$$y_i = \alpha + \beta x_i + \epsilon_i, \quad \epsilon_i \sim \mathrm{N}(0, \sigma^2), \quad i = 1 \ldots n,$$

where $\mathrm{N}(a, b)$ generically denotes a normal distribution with mean a and variance b. Here x_i and y_i are observed and, for simplicity, we assume that σ is known. For a Bayesian analysis of these data, we will assume flat priors on the parameters α and β. The posterior correlation between α and β can then be shown to be

$$\rho_{\alpha\beta} = -\frac{\bar{x}}{\sqrt{\bar{x}^2 + \frac{1}{n}\sum_1^n (x_i - \bar{x})^2}},$$

where $\bar{x} = \frac{1}{n}\sum_1^n x_i$. If $|\bar{x}|$ is large compared to the sample standard deviation of $\{x_i\}$, then $\rho_{\alpha\beta}$ will be close to 1 or minus 1. As discussed above and in Roberts (1995: this volume), high posterior correlations cause poor mixing in Gibbs samplers and in untuned Metropolis–Hastings algorithms.

A simple remedy is to work with centred covariates $x_i' = x_i - \bar{x}$. Then the regression equation becomes $y_i = \alpha' + \beta' x_i' + \epsilon_i$ where $\alpha' = \alpha + \beta\bar{x}$ and $\beta' = \beta$. Note that this has induced a reparameterization $(\alpha, \beta) \rightarrow (\alpha', \beta')$, for which the posterior correlation $\rho_{\alpha'\beta'} = 0$. In this parameterization, the Gibbs sampler will work well. In fact α' and β' are *a posteriori* independent, so the Gibbs sampler will produce samples immediately from the posterior distribution without any burn in. A simple Metropolis algorithm with independent normal proposals for α' and β' will also work well, provided that the proposals are scaled adequately; see Roberts (1995: this volume).

More generally, consider the multiple linear regression model

$$y_i = \theta^{\mathrm{T}} x_i + \epsilon_i, \quad \epsilon_i \sim \mathrm{N}(0, \sigma^2), \quad i = 1 \ldots n,$$

where x_i is a vector of covariates (the first covariate being unity); θ is a vector of parameters with a flat prior; and σ is assumed known. For this model, the posterior variance matrix of θ is $\sigma^2\{x^{\mathrm{T}}x\}^{-1}$, where $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ is the design matrix. If the elements of θ are to be uncorrelated in the posterior, the columns of x must be orthogonal. Orthogonalization with respect to the first column of x is achieved by centring, as described above. For orthogonalizing the whole of x, Gram–Schmidt orthogonalization might be used. However, adequate mixing can be achieved without perfect orthogonalization, since it is usually sufficient to avoid nearly collinear covariates (Hills and Smith, 1992). Also, scaling covariates to roughly equalize sample standard deviations can often help mixing in 'off-the-shelf' Metropolis–Hastings algorithms. The same general recommendations carry over to generalized linear models, with known or unknown scale parameters.

6.2.3 Random-effects models

Gibbs sampling has proved particularly successful for random-effects models, as several chapters in this volume testify. By comparison with fixed-effects models, random-effects models typically contain many parameters, but mixing is usually rapid. However, in some circumstances, mixing can be very poor. To see this, consider the simple random-effects model

$$\begin{aligned} y_{ij} &= \mu + \alpha_i + \epsilon_{ij}, \\ \alpha_i &\sim \mathrm{N}(0, \sigma_\alpha^2), \end{aligned} \tag{6.1}$$

$i = 1, \ldots, m$, $j = 1, \ldots, n$; where $\epsilon_{ij} \sim \mathrm{N}(0, \sigma_y^2)$. For simplicity we suppose σ_α and σ_y are known, and assume a flat prior on μ. Gelfand *et al.* (1995a) show that posterior correlations for this model depend on the relative sizes of the variance components. Posterior correlations for model (6.1) are

$$\rho_{\mu,\alpha_i} = -\left\{1 + \frac{m\sigma_y^2}{n\sigma_\alpha^2}\right\}^{-\frac{1}{2}}, \quad \rho_{\alpha_i,\alpha_j} = \left\{1 + \frac{m\sigma_y^2}{n\sigma_\alpha^2}\right\}^{-1},$$

for $i \neq j$. Large posterior correlations and poor mixing will be avoided if σ_y^2/n is not small in relation to σ_α^2/m. Thus a large number m of random effects or a small random-effects variance σ_α^2 will improve mixing, but a large number n of observations per random effect or small observational variance σ_y^2 will worsen mixing. In short, mixing is worst when the data are most informative!

We now consider two methods for reparameterizing random-effects models.

Reparameterization by hierarchical centring

Gelfand *et al.* (1995a) propose reparameterization to improve mixing in model (6.1). Let $\eta_i = \mu + \alpha_i$, then the reparameterized model is:

$$\begin{aligned} y_{ij} &= \eta_i + \epsilon_{ij}, \\ \eta_i &\sim \mathrm{N}(\mu, \sigma_\alpha^2). \end{aligned} \tag{6.2}$$

With this parameterization, Gelfand *et al.* calculate posterior correlations:

$$\rho_{\mu,\eta_i} = \{1 + \tfrac{mn\sigma_\alpha^2}{\sigma_y^2}\}^{-\frac{1}{2}}, \quad \rho_{\eta_i,\eta_j} = \{1 + \tfrac{mn\sigma_\alpha^2}{\sigma_y^2}\}^{-1}, \tag{6.3}$$

for $i \neq j$. Here, large m or n both improve mixing, and poor mixing will only result if σ_α^2 is very small. Thus, in general, parameterization (6.2) will be preferred since σ_α^2 will not be small if random effects are deemed necessary.

Gelfand *et al.* (1995a) call (6.2) a *centred* parameterization. Note that 'centring' is used here in a quite different sense than in Section 6.2.2; here we are centring parameters, and in Section 6.2.2 we were centring (in a different way) covariates. Gelfand *et al.* extend their approach to nested random-effects models. For example,

$$\begin{aligned} y_{ijk} &= \mu + \alpha_i + \beta_{ij} + \epsilon_{ijk}, \\ \beta_{ij} &\sim \mathrm{N}(0, \sigma_\beta^2), \\ \alpha_i &\sim \mathrm{N}(0, \sigma_\alpha^2), \end{aligned} \tag{6.4}$$

where $i = 1, \ldots, m$; $j = 1, \ldots, n$; $k = 1, \ldots, r$; and $\epsilon_{ijk} \sim \mathrm{N}(0, \sigma_y^2)$. As before, we assume a flat prior on μ and fixed variance components. For this model, the *hierarchically centred* parameterization is $\zeta_{ij} = \mu + \alpha_i + \beta_{ij}$; $\eta_i = \mu + \alpha_i$, so the reparameterized model is:

$$\begin{aligned} y_{ijk} &= \zeta_{ij} + \epsilon_{ijk}, \\ \zeta_{ij} &\sim \mathrm{N}(\eta_i, \sigma_\beta^2), \\ \eta_i &\sim \mathrm{N}(\mu, \sigma_\alpha^2). \end{aligned}$$

Partial centrings are also possible, for example the β parameters might be centred but not the α, giving a (μ, α, ζ) parameterization; or the α parameters might be centered but not the β, giving a (μ, η, β) parameterization. As before, the best of these parameterizations will depend on variance components, and Gelfand *et al.* recommend that effects having large posterior variance relative to σ_y^2 should be centred, along with all effects lower in the hierarchy. If variance components are unknown, the authors suggest using posterior expectations of variance components estimated from a preliminary MCMC run.

Gelfand *et al.* (1995a, 1995b) further generalize their approach to linear and generalized linear mixed-effects models incorporating covariates. With these more general models exact calculations are more difficult, but the recommendation is still to try centred parameterizations.

Reparameterization by sweeping

Vines *et al.* (1995) propose a different approach to reparameterizing random-effects models. They note that model (6.1) is essentially overparameterized. For example, a quantity c could be added to μ and subtracted from each α_i without altering the likelihood of the data. Thus the data are unable to provide any information on the single degree of freedom $\mu - \bar{\alpha}$, where $\bar{\alpha} = \frac{1}{m}\sum_i \alpha_i$. This suggests reparameterizing $\phi_i = \alpha_i - \bar{\alpha}$; $\nu = \mu + \bar{\alpha}$; and $\delta = \mu - \bar{\alpha}$. This is called *sweeping*, since the mean is swept from the random effects and onto μ. This reparameterization gives the model:

$$\begin{aligned} y_{ij} &= \nu + \phi_i + \epsilon_{ij}, \\ \phi_{-m} &\sim \mathrm{N}_{m-1}(0, \sigma_\alpha^2 K_{m-1}), \\ \phi_m &= -\sum_{i=1}^{m-1} \phi_i, \end{aligned} \tag{6.5}$$

where $\phi_{-m} = (\phi_1, \phi_2, \ldots, \phi_{m-1})^{\mathrm{T}}$; N_{m-1} denotes an $(m-1)$-dimensional multivariate normal distribution; and K_{m-1} is an $(m-1) \times (m-1)$ matrix with $1 - \frac{1}{m}$ on the main diagonal and $-\frac{1}{m}$ everywhere else. Clayton (1995: this volume) also discusses this reparameterization.

The random effects $\{\phi_i\}$ are now no longer *a priori* independent, and ν and δ have flat priors. The unidentified degree of freedom δ can be ignored, since the data and other parameters do not depend on it. Note that the original random effects $\{\alpha_i\}$ are interpretable as deviations from the population mean μ, whilst the reparameterized random effects $\{\phi_i\}$ are interpretable as deviations from the sample mean ν.

The *swept* parameterization gives posterior correlations

$$\rho_{\nu,\phi_i} = 0, \quad \rho_{\phi_i,\phi_j} = -\tfrac{1}{m},$$

for $i \neq j$. These correlations are not large for any $m > 1$. They do not depend on n or on variance components σ_α^2 and σ_y^2, unlike those produced by hierarchical centring (6.3).

Vines *et al.* (1995) generalize their approach to models with multiple sets of random effects in generalized linear models, and Vines and Gilks (1994) further extend the approach to accommodate random effects and hierarchical interactions of arbitrary order. The technique is to sweep means from high-order interactions onto lower-order terms. For example, for the nested random-effects model (6.4), the reparameterization proceeds in two stages. First, row means $\bar{\beta}_i = \frac{1}{n}\sum_j \beta_{ij}$ are swept from the $\{\beta_{ij}\}$ parameters onto the $\{\alpha_i\}$; then the mean of $\{\alpha_i + \bar{\beta}_i\}$ is swept onto μ, giving $\psi_{ij} = \beta_{ij} - \bar{\beta}_i$; $\phi_i = \alpha_i - \bar{\alpha} + \bar{\beta}_i - \bar{\beta}$; and $\nu = \mu + \bar{\alpha} + \bar{\beta}$; where $\bar{\beta} = \frac{1}{mn}\sum_i \sum_j \beta_{ij}$. This reparameterization gives the model:

$$\begin{aligned}
y_{ijk} &= \nu + \phi_i + \psi_{ij} + \epsilon_{ijk}, \\
\psi_{i,-n} &\sim \mathrm{N}_{n-1}\left(0, \sigma_\beta^2 K_{n-1}\right), \\
\psi_{i,n} &= -\sum_{j=1}^{n-1} \psi_{ij}, \\
\phi_{-m} &\sim \mathrm{N}_{m-1}\left(0, \sigma_\alpha^2 + \frac{1}{n}\sigma_\beta^2 K_{m-1}\right),
\end{aligned}$$

where $\psi_{i,-n} = (\psi_{i1}, \psi_{i2}, \ldots, \psi_{i,n-1})^{\mathrm{T}}$, and ν has a flat prior.

Unknown variance components

The above reparameterizations can reduce posterior correlations between random effects even if variance components are unknown. However, lack of information about variance components can itself be a source of slow mixing. Moreover, when this problem arises it seems difficult to cure. In the simple random effects model (6.1), slow mixing tends to occur if the prior on the random-effects variance σ_α^2 gives non-negligible probability to values near zero. An extreme form of this is the improper prior $P(\sigma_\alpha^2) \propto \sigma_\alpha^{-2}$, which gives an improper posterior (see DuMouchel and Waternaux, 1992). Then the MCMC sampler will get stuck for long periods where σ_α^2 and the sample variance of random effects $\frac{1}{m}\sum(\alpha_i - \bar{\alpha})^2$ are both small. At present, the only remedy we can suggest is to use a different prior, perhaps bounding σ_α^2 away from zero.

6.2.4 Nonlinear models

It is difficult to lay down hard and fast rules for reparameterization with nonlinear models. However, much of the experience gained in dealing with such models in the context of maximum likelihood (see for example Ross, 1990) or numerical quadrature (see Hills and Smith, 1992, and references therein) is still of relevance for MCMC, since the aim is often to reparameterize to produce open, independent-normal-like contours as illustrated in Figure 6.1(b).

The advice of Ross (1990) is to reparameterize so that parameters correspond approximately to contrasting features of the data. These are called *stable* parameters. For example, consider the model

$$E(y_i) = \alpha + \beta e^{-\gamma x_i},$$

$i = 1, \ldots, m$. If the data are approximately linear, γ will be small. Then $E(y_i) \approx \alpha + \beta - \beta\gamma x_i$, giving high posterior correlations between the

parameters. A stable parameterization would be

$$E(y_i) = \alpha' + \beta'[f_i(\gamma) - \bar{f}(\gamma)],$$

where

$$\begin{aligned}
f_i(\gamma) &= -\frac{1}{\gamma x^*} e^{-\gamma(x_i - x^*)}, \\
\bar{f}(\gamma) &= \frac{1}{m}\sum f_i(\gamma),
\end{aligned}$$

and x^* is some central value of x. Now for small γ, $E(y_i) \approx \alpha' + \beta'(x_i - \bar{x})$, as motivated in Section 6.2.2. The stable parameters α' and β' correspond approximately to the mean height of the curve and the slope at x^*.

In general, divination of stable parameterizations must be done on a model-by-model basis, although similarity with well understood models can help. See Ross (1990) for further suggestions; see also Bennett *et al.* (1995: this volume) for an application of MCMC to nonlinear models.

6.2.5 General comments on reparameterization

We have indicated some strategies for reparameterizing commonly used statistical models to reduce posterior correlations. More ambitious strategies aim to reparameterize on the basis of posterior correlations estimated from the output from a preliminary MCMC run (Müller, 1994; Hills and Smith, 1992; Wakefield, 1992). However, the performance of these methods in high dimensions is unknown. Moreover, such methods may have substantial computational overheads, partly through destroying conditional-independence relationships present in the original parameterization.

Conditional-independence relationships allow high-dimensional problems to be tackled efficiently using the Gibbs sampler or a single-component Metropolis–Hastings algorithm (Spiegelhalter *et al.*, 1995b: this volume). For example, the full conditional distribution for α_i in model (6.1) is algebraically independent of $\{y_{kj}; k \neq i, j = 1, \ldots, n\}$ and can therefore be rapidly calculated. 'Natural' parameterizations, which maintain a healthy conditional-independence structure, are therefore advantageous.

Models often admit several natural parameterizations, as illustrated in Section 6.2.3. In the absence of knowledge about which will produce rapid mixing, all might be tried in a random or cyclic order (Gilks, 1995a). The combined strategy will mix well if at least one parameterization mixes well (Tierney, 1994).

For roughly normal or log-concave posterior distributions, the avoidance of high posterior correlations will often be sufficient to produce good mixing (Hills and Smith, 1992). However, other forms of posterior can yield poor mixing even in the absence of posterior correlations. For example, the Metropolis algorithm can be non-geometrically ergodic with heavy-tailed distributions (see Roberts, 1995: this volume). Hills and Smith (1992) suggest a signed root transformation to help in such situations. Another cause of slow mixing is multimodality in $\pi(.)$. Reparameterization may not help here, and we may need to turn to other strategies.

6.3 Random and adaptive direction sampling

The Gibbs sampler moves in directions parallel to the coordinate axes and, as we have seen, these directions may not be conducive to rapid mixing. Linear reparameterization is equivalent to changing the directions in which the Gibbs sampler moves. For example, Gibbs sampling with the reparameterization described in Section 6.2.1 is equivalent to Gibbs sampling in directions $X_{.2} = X_{.1}$ and $X_{.2} = -X_{.1}$ in the original parameterization. When it is not clear how to choose sampling directions to produce rapid mixing, a solution might be to construct a sampler which can sample in any direction. In this section, we describe some such samplers, the simplest being the ***hit-and-run*** algorithm.

6.3.1 The hit-and-run algorithm

The hit-and-run algorithm (Schmeiser and Chen, 1991) is a MCMC sampler which chooses sampling directions in $\mathbb{R}^k$ at random. At each iteration t, the hit-and-run algorithm first randomly samples a direction e_t (a unit vector of dimension k). Then X_{t+1} is chosen according to the full conditional distribution along the straight line passing through X_t in direction e_t. Thus each hit-and-run iteration comprises:

Step 1: sample e_t;
Step 2: sample a scalar r_t from density $f(r) \propto \pi(X_t + re_t)$;
Step 3: set $X_{t+1} = X_t + r_t e_t$.

Often, but not necessarily, e_t is chosen uniformly on the unit k-dimensional sphere. This may be done by generating k independent standard normal random variates $\{z_i;\ i = 1, \ldots, k\}$, and then setting

$$e_{t.i} = \frac{z_i}{\sqrt{\sum_j z_j^2}},$$

for $i = 1, \ldots, k$.

Under extremely weak regularity conditions, the hit-and-run algorithm is irreducible. Moreover, it can often mix better than the Gibbs sampler. Consider for example the target density in Figure 6.1(a). Most of the sampling directions generated by the hit-and-run algorithm will be away from the diagonal line $X_{.2} = X_{.1}$, and consequently X_{t+1} will tend to lie near X_t. Occasionally, however, a sampling direction close to the diagonal line will be generated, and then X_{t+1} will have the opportunity to move well away

from X_t. This would not happen for the Gibbs sampler without reparameterization. The hit-and-run algorithm can also work well in multimodal problems, allowing mode-hopping when the sampling direction traverses two or more modes of $\pi(.)$. The algorithm can be especially useful as an exploratory tool for discovering effective sampling directions.

If $\pi(.)$ has moderately sharp ridges or spikes, random directions may not pick out regions of high probability content sufficiently often to promote rapid mixing. These problems are exacerbated in high dimensions, where the method may perform poorly.

6.3.2 Adaptive direction sampling (ADS)

ADS (Gilks *et al.*, 1994; Roberts and Gilks, 1994) can be thought of as a generalization of the hit-and-run algorithm. It is an *adaptive* method in the sense that the direction to be sampled, e_t, can be chosen on the basis of other previously sampled points.

In general, care must be taken when using adaptive methods since the stationarity of the target distribution $\pi(.)$ can be compromised by adaptation (see for example Gelfand and Sahu, 1994). ADS avoids this problem by enlarging the Markov chain space to allow points on which adaptation is based to be part of the Markov chain state vector.

Specifically, at each iteration we store m points $\{X_t^{(1)}, X_t^{(2)}, \ldots, X_t^{(m)}\}$. These m points are called the *current set*. Each point of the current set is a k-vector in its own right. At each iteration, a point $X_t^{(c)}$ is randomly selected from the current set, and is updated. Its new value is sampled along a line passing through $X_t^{(c)}$ in a direction e_t (not necessarily a unit vector) which may depend on any or all of the points in the current set. The remaining points in the current set are not changed. Thus each iteration of ADS comprises:

Step 1: select one point $X_t^{(c)}$ at random from current set;
Step 2: choose a k-vector e_t;
Step 3: sample a scalar r_t from a density $f(r)$;
Step 4: set $X_{t+1}^{(c)} = X_t^{(c)} + r_t e_t$;
Step 5: set $X_{t+1}^{(i)} = X_t^{(i)}$ for $i \neq c$.

With the general framework for constructing direction e_t and density $f(.)$ described below, the stationary distribution for ADS is the distribution of m independently sampled points from the target density $\pi(.)$. The simplest special case of ADS is the hit-and-run algorithm, where $m = 1$, and e_t and r_t are sampled as described in Section 6.3.1.

A more interesting special case of ADS is the *snooker algorithm*. For this, the sampling direction is $e_t = X_t^{(i)} - X_t^{(c)}$, where $X_t^{(i)}$ is a second point chosen at random from the current set; and the density

$$f(r) \propto \pi(X_t^{(c)} + re_t)|1-r|^{k-1}.$$

Thus $X_{t+1}^{(c)}$ lies along the straight line passing through $X_t(c)$ and $X_t(i)$. This is illustrated in Figure 6.2. Heuristically, the idea of the snooker algorithm is that, as the algorithm proceeds, sampling directions are increasingly likely to traverse regions of high probability under $\pi(.)$, since both $X_t^{(c)}$ and $X_t^{(i)}$ will increasingly tend to be in such regions. Note that $f(r)$ is just the full conditional density along the line $X_t^{(c)} + re_t$, multiplied by $|1-r|^{k-1}$. This multiplier derives from a Jacobian matrix: it arises because the sampling direction for updating $X_t^{(c)}$ depends on $X_t^{(c)}$ itself, unlike the hit-and-run algorithm and Gibbs sampler.

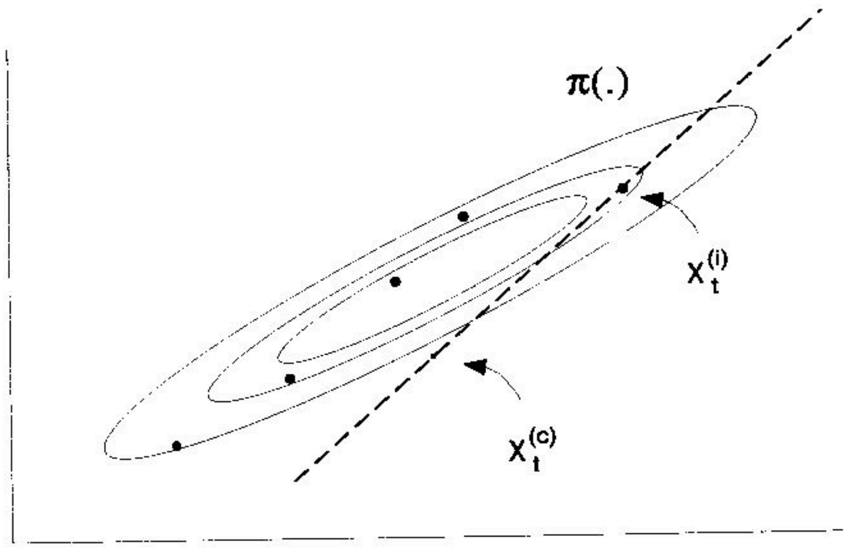

Figure 6.2 *Illustrating the snooker algorithm. Ellipses denote contours of the target density $\pi(.)$; dots denote points in the current set; and the broken line is the sampling direction for updating $X_t^{(c)}$.*

The snooker algorithm has some desirable theoretical properties, and can work well in problems where the Gibbs sampler fails badly. However, the increased computational burden it demands, and its inconsistent performance on some problems that simpler algorithms solve easily, suggest the need for schemes which make more productive use of the adaptive information contained in the current set.

For the general form of ADS, the sampling direction is $e_t = u_t X_t^{(c)} + V_t$, where u_t is a random scalar drawn independently of the current set, and

V_t is a random k-vector depending on the $m-1$ points $\{X_t^{(i)}; i \neq c\}$. Then

$$f(r) \propto \pi(X_t^{(c)} + re_t)|1+ru_t|^{k-1}.$$

ADS can be generalized still further to Metropolized versions (*Adaptive Metropolis Sampling*, AMS). Like ADS, AMS updates a randomly chosen point $X_t^{(c)}$ from a current set of m points, but does this using a Metropolis–Hastings step, where the proposal for a new current set $\{Y^{(i)}; i = 1, \ldots, m\}$ has density $q(\{X_t^{(i)}\}; \{Y^{(i)}\})$, and depends on any or all of the points in the current set $\{X_t^{(i)}; i = 1, \ldots, m\}$. The acceptance probability is

$$\min\left\{1, \frac{q(\{Y^{(i)}\}; \{X_t^{(i)}\}) \prod_{i=1}^m \pi(Y^{(i)})}{q(\{X_t^{(i)}\}; \{Y^{(i)}\}) \prod_{i=1}^m \pi(X_t^{(i)})}\right\}$$

The stationary distribution for AMS is the distribution of m independently sampled points from the target density $\pi(.)$. As for ADS, successful implementation of AMS requires effective use of the adaptive information contained in the current set.

AMS provides a general framework for adaptation where the information from which the adaptation is based depends on only a finite history (the m points of the current set). Adaptive methods where the whole history of the chain is taken into account cannot be considered in this way. It is also doubtful that *purely* adaptive methods are advisable. Adaptation to the 'wrong' information, obtained early on in the MCMC, might worsen mixing. We might then never discover that the early information was misleading. Mathematically, this problem manifests itself in that many adaptive algorithms are not *geometrically convergent* (see Roberts, 1995: this volume). It is always advisable when using adaptive methods to use the adaptive strategy in hybridization with some fixed non-adaptive strategy.

6.4 Modifying the stationary distribution

This section describes techniques which aim to improve mixing by modifying the stationary distribution $\pi(.)$ of the Markov chain.

6.4.1 Importance sampling

Suppose we can devise a MCMC sampler which mixes well but has a modified stationary distribution $\pi^*(.)$, where $\pi^*(X) \approx \pi(X)$ for all X. We can estimate the expectation under $\pi(.)$ of an arbitrary function $g(X)$ of interest by importance reweighting the output $\{X_t;\ t = 1, \ldots, n\}$ from the chain with stationary distribution $\pi^*(.)$ (Fosdick, 1963; Hastings, 1970). Thus,

$$E_\pi g(X) \approx \frac{\sum_t w_t g(X_t)}{\sum_t w_t} \tag{6.6}$$

where the importance weight $w_t = \pi(X_t)/\pi^*(X_t)$. See, for example, Ripley (1987) for further discussion of importance sampling.

Jennison (1993) suggests using the form $\pi^*(X) = \pi(X)^{\frac{1}{T}}$, where $T > 1$. This form of modification of the target $\pi(.)$ is used in the optimization technique of simulated annealing, where T is called *temperature*. Heating the target ($T > 1$) will flatten $\pi(.)$ and may make it easier for the modified MCMC sampler to explore the support of $\pi(.)$. For example, suppose $\pi(.)$ is multimodal with well-separated modes which contain most of the probability content of $\pi(.)$. For this $\pi(.)$, a Metropolis algorithm proposing mainly local moves will mix poorly, since the chain will be trapped for long periods at one mode before escaping to another mode. Heating the target will flatten the modes and place more probability between them, so the modified MCMC sampler will travel more easily between modes. Note that this $\pi^*(.)$ is unnormalized, but this presents no problem for MCMC methods.

Another useful form of modification of the target distribution is to add stepping stones between regions of the state-space which contain substantial amounts of $\pi(.)$ but which do not communicate well (i.e. between which the MCMC sampler seldom moves). This is illustrated in Figure 6.3, in which the support D of $\pi(.)$ is concentrated in two disjoint regions. The Gibbs sampler (or a single-component Metropolis–Hastings algorithm making moves parallel to the axes) applied to this problem would be reducible, since moving between the two regions would entail an intermediate move X_t outside D, where $\pi(X_t) = 0$. This problem can be avoided by placing a stepping stone E, containing a small amount of probability ϵ, such that E communicates with both regions of D as illustrated in Figure 6.3. Then $\pi^*(X) \propto \pi(X) + \epsilon e(X)$, where $e(X)$ is a density having support on E. If D and E are disjoint, importance reweighting as in (6.6) dictates that output samples which belong to E should be ignored. This technique can be useful in problems which place intricate constraints on X, as can occur for example in genetics applications (Sheehan and Thomas, 1993; see also Thomas and Gauderman, 1995: this volume).

In general, if $\pi^*(X)$ differs substantially from $\pi(X)$, importance weights will be unstable and the variance of (6.6) will be inflated, which may necessitate lengthening the MCMC run. Therefore, the success of the method relies on being able to obtain a rapidly mixing MCMC with only slight modification of the stationary distribution $\pi(.)$. This may be unrealistic for problems where mixing is very slow. However, importance reweighting of MCMC output is useful for other reasons, for example for exploring the impact of small changes in the prior or likelihood of a model (Besag *et al.*, 1995).

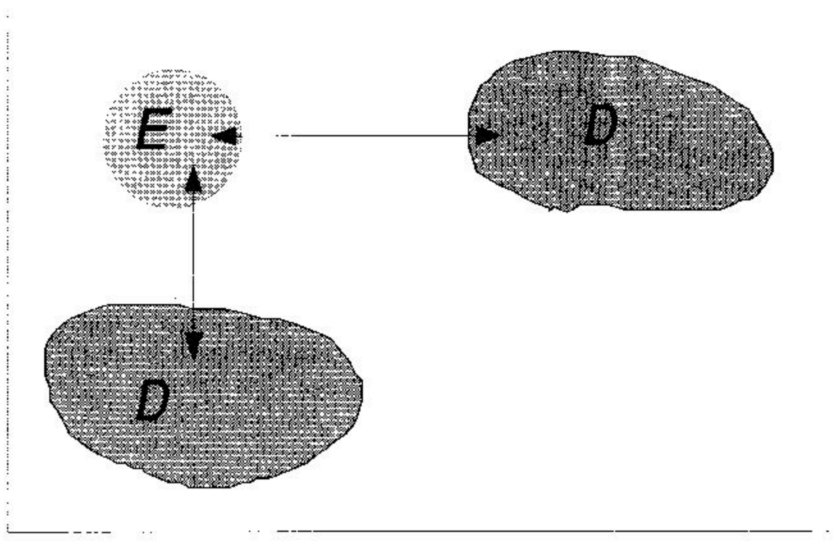

Figure 6.3 *Adding a stepping stone E to allow communication between disjoint regions D of the support of $\pi(.)$, as indicated by the arrows.*

6.4.2 *Metropolis-coupled MCMC*

Geyer (1991) proposes running in parallel m MCMC chains with different stationary distributions $\pi_i(X)$, $i = 1, \ldots, m$, where $\pi_1(X) = \pi(X)$ and $\{\pi_i(X);\ i > 1\}$ are chosen to improve mixing. For example, incremental heating of the form

$$\pi_i(X) = \pi(X)^{\frac{1}{1+\lambda(i-1)}}, \qquad \lambda > 0,$$

is often convenient, but not always desirable; see Geyer and Thompson (1995). After each iteration an attempt is made to swap the states of two of the chains using a Metropolis–Hastings step. Let $X_t^{(i)}$ denote the state of chain i at iteration t. Suppose that after iteration t a swap between chains i and j is proposed. The probability of accepting the proposed swap is

$$\min\left\{1, \frac{\pi_i(X_t^{(j)})\pi_j(X_t^{(i)})}{\pi_i(X_t^{(i)})\pi_j(X_t^{(j)})}\right\}.$$

At the end of the run, output from the modified chains $\{X_t^{(i)};\ t > 0, i > 1\}$ can be discarded. This method is called *Metropolis-coupled* MCMC (or MCMCMC).

Heuristically, swapping states between chains will confer some of the rapid mixing of the modified chains upon the unmodified chain. For example, suppose that $\pi(.)$ has well separated modes and that a modified chain moves freely between modes. Proposing swaps between this modified chain and the unmodified chain will sometimes result in the unmodified chain changing modes, thereby improving mixing. Proposed swaps will seldom be accepted if $\pi_i(X)/\pi_j(X)$ is very unstable; this is the reason for using several chains which differ only gradually with i.

An obvious disadvantage of Metropolis-coupled MCMC is that m chains are run but the output from only one is utilized. However, Metropolis-coupled MCMC is ideally suited to implementation on a parallel processing machine or even on a network of workstations (each processor being assigned one chain), since each chain will in general require about the same amount of computation per iteration, and interactions between chains are simple.

6.4.3 *Simulated tempering*

An idea closely related to Metropolis-coupled MCMC is *simulated tempering* (Marinari and Parisi, 1992; Geyer and Thompson, 1995). Instead of running in parallel m MCMC samplers with different stationary distributions, they are run in series and randomly interchanged. Thus simulated tempering produces one long chain, within which are embedded variable length runs from each sampler and occasional switches between samplers. The term 'simulated annealing' arises from an analogy with repeated heating and cooling of a metal.

Let $\pi_i(.)$ denote the stationary distribution of the i^{th} sampler, where $i = 1$ denotes the unmodified (cold) sampler, and let I_t indicate which sampler is current at iteration t of the simulated tempering chain. The state of the chain at time t comprises the pair (X_t, I_t). One iteration of simulated tempering involves an update of the current point X_t using sampler I_t, followed by a Metropolis–Hastings update of I_t.

Consider now the Metropolis–Hastings update of I_t. Let $q_{i,j}$ denote the probability that sampler j is proposed given that the current sampler is the i^{th}. Geyer and Thompson (1995) suggest setting $q_{i,i+1} = q_{i,i-1} = 0.5$ if $1 < i < m$ and $q_{1,2} = q_{m,m-1} = 1$, so that only adjacent samplers are proposed. Then the proposal to change to sampler j is accepted with probability

$$\min\left\{1, \frac{c_j\pi_j(X_t)q_{j,i}}{c_i\pi_i(X_t)q_{i,j}}\right\}, \tag{6.7}$$

where $i = I_t$ and the $\{c_i;\ i = 1, \ldots, m\}$ are conveniently chosen constants (see below).

The stationary distribution of the simulated tempering chain is $\pi(X, I) \propto c_I\pi_I(X)$. Since we are usually interested only in $\pi_1(.) = \pi(.)$, at the end of the run all samples for which $I_t \neq 1$ are normally discarded. After an initial burn-in, all samples $\{X_t\}$ for which $I_t = 1$ are retained; there is no need for a new burn-in after each change of sampler.

The constants $\{c_i;\ i = 1, \ldots, m\}$ are chosen so that the chain divides its

time roughly equally among the m samplers. If the $\pi_i(.)$ were normalized, this would be achieved by setting $c_i = 1$ for all i. However, $\pi_i(.)$ will usually be known only up to a normalizing constant, so it will be necessary to set $c_i \approx 1/\int \pi_i(X)dX$. Thus simulated tempering requires estimation of normalizing constants, unlike Metropolis-coupled MCMC. Precise estimation of the $\{c_i\}$ is not required for valid analysis of $\pi(.)$, but even approximate estimation of the $\{c_i\}$ can be tricky. Geyer (1993) suggests several *ad hoc* techniques, the most formal of which is *reverse logistic regression*.

Reverse logistic regression (Geyer, 1993) requires a preliminary run of Metropolis-coupled MCMC which at least mixes a little. The $\{c_i\}$ are then estimated by maximizing with respect to $\{c_i\}$ a log quasi-likelihood

$$l_n(c) = \sum_t \sum_{i=1}^{m} \log p_i(X_t^{(i)}, c)$$

in the notation of Section 6.4.2, where

$$p_i(X, c) = \frac{c_i\pi_i(X)}{\sum_{j=1}^{m} c_j\pi_j(X)}.$$

An arbitrary constraint on the $\{c_i\}$ is required to identify the optimization. Note that reverse logistic regression 'forgets' the sample i from which each $X_t^{(i)}$ belongs.

Geyer and Thomson (1995) suggest setting the number of samplers m and the 'spacing' of their stationary distributions (for example, the temperature spacing λ) so that average acceptance rates in (6.7) are between 20% and 40%. This range of acceptance rates is also recommended for Metropolis algorithms in other settings (Gelman *et al.*, 1995; Roberts, 1995: this volume). See Geyer (1993) and Geyer and Thomson (1995) for further implementational suggestions.

If the m^{th} sampler draws samples independently from $\pi_m(.)$, the simulated tempering chain will 'forget' its past (i.e. *regenerate*) whenever sampler m is used. This is the basis of *regenerative simulation* techniques (Mykland *et al.*, 1995; see also Tierney, 1995: this volume). In particular, sample paths between regenerations are independent so Monte-Carlo standard errors can be estimated easily. Also, there is no need for an initial burn-in if the chain is begun with a draw from sampler m.

6.4.4 *Auxiliary variables*

Adding variables can often simplify calculations and lead to improved mixing. For example, when some data x_m are missing, it may be difficult to work with the marginal posterior density $\pi(\theta) \propto P(\theta)\int P(x_o, x_m|\theta)dx_m$ of the model parameters given the observed data x_o, particularly if the marginal is not available in closed form. In general, it will be far simpler to run the MCMC on an augmented state vector (θ, x_m) and a full posterior $\pi^*(\theta, x_m) \propto P(\theta)P(x_o, x_m|\theta)$, sampling both missing data and model parameters in the MCMC. An important example occurs in survival analysis, where x_m is the set of true (but unknown) failure times for censored individuals.

Added variables are called ***auxiliary variables*** by Besag and Green (1993), and the idea of running a MCMC on an augmented state vector is the essence of ***data augmentation*** (Tanner and Wong, 1987). Auxiliary variables U need not have an immediate interpretation such as missing data. The task is to choose a convenient conditional density $P(U|X)$ and an MCMC sampler which samples both X and U with stationary distribution $\pi^*(X, U) = \pi(X)P(U|X)$, such that the resulting chain is rapidly mixing. Note that the marginal distribution for X is $\pi(X)$, as required. Note also that, in the Bayesian context, $\pi(X)$ implicitly conditions on the observed data; therefore $\pi(U|X)$ may also condition on the observed data. Having run the MCMC, the $\{X_t\}$ samples can be used for inference about $\pi(X)$, and the $\{U_t\}$ samples can be ignored.

The most famous example of the use of auxiliary variables is the ***Swendsen–Wang algorithm*** (Swendsen and Wang, 1987), which finds application in certain lattice models; this is discussed in Green (1995: this volume). Some of the strategies discussed above can be viewed as auxiliary variables methods, as can adaptive rejection Metropolis sampling (Gilks *et al.*, 1995a; Gilks, 1995b: this volume).

Here we somewhat speculatively suggest another auxiliary variable strategy, for use when the target density $\pi(.)$ can be approximated by a more convenient density $\pi_0(.)$ for most X, but not everywhere. For example, $\pi_0(.)$ might be a multivariate normal approximation to the target density, permitting independent sampling. Another example might be a posterior density having a simple form apart from an awkward multiplicative term arising perhaps through an ascertainment mechanism. Such terms often make little difference for moderately probable parameter values, but typically involve numerical integration. Here $\pi_0(.)$ would be the posterior without the awkward multiplier.

As usual we do not need to assume that $\pi(.)$ and $\pi_0(.)$ are normalized densities. Define a region A such that $\pi(X) \approx \pi_0(X)$ for all $X \in A$; see Figure 6.4. Let c be a constant such that $c \leq \pi(X)/\pi_0(X)$ for all $X \in A$. Region A might be chosen to make the calculation of c easy. Define a scalar auxiliary variable U taking values 0 and 1, with

$$P(U = 1|X) = \begin{cases} c\frac{\pi_0(X)}{\pi(X)} & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases} . \tag{6.8}$$

Note that c ensures $P(U = 1|X) \leq 1$. Two different MCMC samplers S_1 and S_0 are employed for updating X; S_1 having stationary density propor-

tional to $\pi_0(X)I(X \in A)$, and S_0 having stationary density proportional to $\pi(X) - c\pi_0(X)I(X \in A)$. Here $I(.)$ is the indicator function, taking the value 1 when its argument is true, and zero otherwise. Each iteration of the algorithm then proceeds as follows:

Step 1: If $U_t = 1$ sample X_{t+1} using sampler S_1
else sample X_{t+1} using sampler S_0;

Step 2: Sample U_{t+1} directly from (6.8).

This chain has the required stationary distribution $\pi^*(X, U)$.

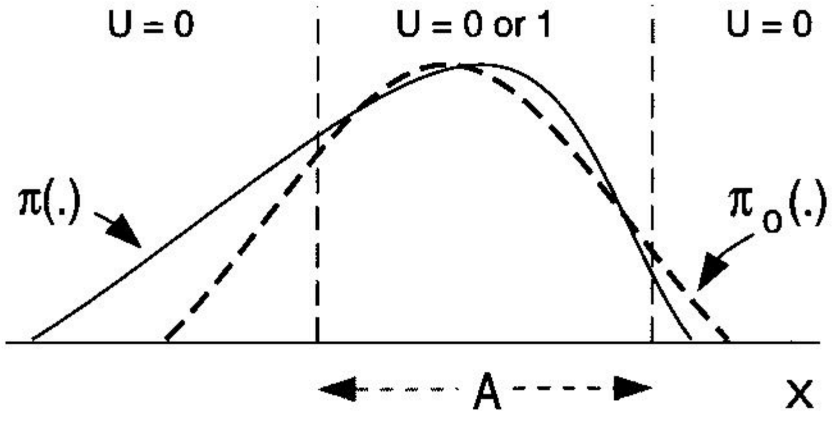

Figure 6.4 *Illustrating the use of an auxiliary variable U; $\pi_0(X)$ approximates $\pi(X)$ in region A. See text for explanation.*

If independent sampling from $\pi_0(.)$ is possible, S_1 might sample independently from $\pi_0(.)$, rejecting any samples falling outside A; and S_0 might be a Metropolis algorithm. If A and c are chosen well, the chain should spend most of its time with $U_t = 1$, where mixing is rapid. When $U_t = 0$, the chain may mix less well, but careful choice of S_0 may help prevent the chain from getting stuck in one place for many iterations. Whenever $U_t = 1$, the chain *regenerates*, and techniques of regenerative simulation can be employed to estimate Monte-Carlo standard errors (Mykland *et al.*, 1995; Tierney, 1995: this volume; see also Section 6.4.3 below).

As noted above, $\pi(X_t)$ may be particularly expensive to compute. While $U_t = 1$, $\pi(X_t)$ need not be evaluated at Step 1 since sampler S_1 needs to evaluate only $\pi_0(X_t)$. However, $\pi(X_t)$ must be evaluated in Step 2. If an upper bound $d \geq \pi(X)/\pi_0(X)$ for all $X \in A$ is available, considerable computational savings can be made in Step 2. According to (6.8), if $X_t \notin A$, we must have $U_{t+1} = U_t = 0$. If $X_t \in A$, Step 2 can be performed by:

Step 2a: Set $U_{t+1} = 1$ with probability $\frac{c}{d}$;

Step 2b: If now $U_{t+1} = 0$, reset $U_{t+1} = 1$ with probability $c\frac{\pi_0(X_t)}{\pi(X_t)} - \frac{c}{d}$.

If c and d squeeze $\pi(X)/\pi_0(X)$ tightly within region A, U_{t+1} will usually be set to 1 at Step 2a, and $\pi(X_t)/\pi_0(X_t)$ will not need to be evaluated. Thus if the chain spends most of the time with $U_t = 1$, good mixing will be achieved with minimal computation.

The above techniques can be readily generalized to single-component Metropolis–Hastings samplers, so that for each full conditional distribution $\pi(. \mid \{X_{t.j}; j \neq i\})$ an approximation $\pi_0(. \mid \{X_{t.j}; j \neq i\})$ is devised, and an auxiliary variable U_i is constructed analogously to (6.8).

6.5 Methods based on continuous-time processes

There are a number of MCMC methods derived from the properties of continuous-time Markov processes which can be used to simulate from target densities. See Chapter 5 of Neal (1993) for a review of many of these methods and their motivations from a physical perspective.

First, we consider a k-dimensional diffusion process (a continuous-time, continuous-sample-path Markov chain in k dimensions) which has a stationary distribution $\pi(.)$. Diffusion processes are best described by *stochastic differential equations* of the form

$$dX_t = \Sigma^{\frac{1}{2}} dB_t + \mu(X_t)dt \tag{6.9}$$

where dX_t is the change $X_{t+dt} - X_t$ in state vector X occuring in an infinitessimally small interval of time dt; Σ is a constant positive-definite $k \times k$ matrix; μ is a vector-valued function of length k; and dB_t is an increment of k-dimensional *Brownian motion*. Loosely speaking, (6.9) says that, given X_t,

$$X_{t+dt} \sim \mathrm{N}_k(X_t + \mu(X_t)dt, \Sigma dt), \tag{6.10}$$

where N_k denotes a k-dimensional multivariate normal density.

If we set

$$\mu(X) = \frac{1}{2}\nabla \log \pi(X), \tag{6.11}$$

where ∇ denotes $(\frac{\partial}{\partial X_{.1}}, \frac{\partial}{\partial X_{.2}}, \ldots, \frac{\partial}{\partial X_{.k}})^{\mathrm{T}}$, and if $\Sigma = I_k$ (the identity matrix) then, under suitable regularity conditions (see for example Roberts and Tweedie, 1995), $\pi(.)$ is the unique stationary and limiting distribution for the process X, and X is called the *Langevin* diffusion for $\pi(.)$. From now on, we shall assume that (6.11) holds.

Thus, if we were able to run this continuous-time process, at each time t after a suitable burn-in, the sample X_t could be regarded as a sample from the target density $\pi(.)$. In practice of course, simulation from diffusions is impossible and a discrete-time approximation is necessary. A natural choice

is suggested by (6.10):

$$X_{t+\delta} \sim \mathrm{N}_k(X_t + \delta\mu(X_t), \delta I_k), \tag{6.12}$$

for some small $\delta > 0$. Neal (1993) discusses the interpretation of this algorithm in terms of Hamiltonian dynamics, the use of correlated noise, and issues related to time discretization.

Unfortunately, the time discretization can have dire consequences for the stationary distribution of the algorithm. In particular, it is quite common to find that the algorithm is not even recurrent, no matter how fine the time discretization δ (Roberts and Tweedie, 1995). To counter this, it is advisable to use a correcting Metropolis–Hastings rejection step. Therefore, given X_t, the Langevin–Hastings algorithm proceeds by generating a proposal X' from $\mathrm{N}_k(X_t+\delta\mu(X_t), \delta I_k)$, and accepting this proposal (setting $X_{t+\delta} = X'$) with probability the minimum of 1 and

$$\frac{\pi(X')}{\pi(X_t)} \exp\left\{-\frac{1}{2}[\mu(X') + \mu(X_t)]^{\mathrm{T}}[2(X' - X_t) + \delta\{\mu(X') - \mu(X_t)\}]\right\}.$$

This correction preserves the stationarity of $\pi(.)$, but the algorithm produced is very often non-geometric (Roberts and Tweedie, 1995).

Recent work by Hwang *et al.* (1993) has demonstrated that, for Gaussian target densities, the Langevin scheme described above can be improved upon by a non-reversible diffusion process.

The Langevin–Hastings algorithm can be considered as an alternative to the random-walk Metropolis algorithm, where the proposal distribution is adjusted by considering a local property (the gradient of $\log \pi(.)$). This adjustment tends to propose points in the uphill direction from X_t. Thus the chain is nudged in the direction of modes. If the local properties of $\pi(.)$ are erratic however, this nudge can be misleading, and could for example result in wildly overshooting modes. Therefore, care has to taken when using this method, to check that the target density is sufficiently smooth, and has no zeros in the interior of the state-space. Alternatively, algorithms with truncated proposals of the form

$$\mathrm{N}_k(X_t + \min\{b, \delta\mu(X_t)\}, \delta I_k)$$

for large fixed b, retain most of the problem specific advantages of the Langevin–Hastings algorithm, without the potential inherent instability of the algorithm.

Extensions of this approach include the use of jump-diffusions, perhaps allowing the Markov chain to jump between different-dimensional subspaces of the parameter space, whilst following a diffusion sample path within each subspace. See Phillips and Smith (1995: this volume) and Grenander and Miller (1994) for applications.

6.6 Discussion

Our discussion to this point has been focused mainly on improving mixing. We have said little about reducing Monte-Carlo standard errors. Perfect mixing is achieved by independent sampling from $\pi(.)$, but a suitable choice of proposal can reduce variances below those of independent sampling (Peskun, 1973). To reduce Monte-Carlo standard errors in single-component Metropolis–Hastings samplers, Besag and Green (1993) suggest using proposal distributions which incorporate an antithetic effect, to encourage $X_{t.i}$ to flip from one side of its full conditional density to the other. Importance reweighting of samples generated from a distribution with heavier tails than $\pi(.)$ can also reduce Monte-Carlo standard errors (Geweke, 1989). The performance of these strategies may be sensitive to the functionals $g(.)$ of interest, but a well mixing sampler will generally deliver small Monte-Carlo standard errors for all functionals of interest. Small Monte-Carlo standard errors may be achieved most easily at the expense of rapid mixing; zero variances are guaranteed with a proposal which sets $X_{t+1} = X_t$ with probability 1.0! Thus we regard rapid mixing as a higher ideal than variance reduction.

In this chapter we have reviewed a variety of strategies for reducing run times. None are guaranteed to achieve reductions, as all are justified to some extent heuristically. However, an appreciation of the source of the problem in a given application may suggest which strategy might improve the MCMC. We recommend first trying some reparameterizations, or equivalently some carefully tailored Metropolis–Hastings proposal densities. If these do not help, one of the techniques of Section 6.4 might. If still no progess has been made, there is enormous scope for experimentation within the frameworks outlined in Sections 6.3 and 6.5.

The possibilites do not end there. Markov chain mixing can be improved by *mixing strategies* within a single chain (Tierney, 1994), i.e. using different strategies at different iterations of the chain (note two different uses of the word 'mixing' here). For example, different reparameterizations and auxiliary variables might be used at different iterations. As another example, the blocking scheme (which defines the components in a single-component sampler) can be altered as the simulation proceeds; this is the essence of *multigrid methods* (see Goodman and Sokal, 1989, for an extensive review). When mixing strategies, it is important to ensure that the choice of strategy at iteration t does not depend on X_t, to avoid disturbing the stationary distribution of the chain.

MCMC methodology has had a profound effect on liberating statistical modelling, and application of sophisticated models has been made feasible through specially developed software, in particular BUGS (Spiegelhalter *et al.*, 1995a). Having adjusted to the new freedom, many practitioners are now working at the operational limits of existing MCMC methodology, the limits essentially being met in the form of exasperatingly slow mixing. Thus there is an urgent need to develop methods for improving mixing for ever greater levels of model generality. Without very general and robust methods, reliable general-purpose software will not be attainable, and without such software, much that MCMC has to offer will be inaccessible to the greater corpus of applied statisticians.

A remarkable feature of the general framework of MCMC is the scope it affords for ingenuity and creativity in developing rapidly mixing chains for classes of problem. The techniques reviewed in this chapter give some indication of the possibilities. Undoubtedly there will be many exciting new developments in the future.

Acknowledgement

We are grateful to Nicky Best for many helpful comments on an earlier version of this chapter.

References

Bennett, J. E., Racine-Poon, A. and Wakefield, J. C. (1995) MCMC for nonlinear hierarchical models. In *Markov Chain Monte Carlo in Practice* (eds W. R. Gilks, S. Richardson and D. J. Spiegelhalter), pp. 339–357. London: Chapman & Hall.

Besag, J. and Green, P. J. (1993) Spatial statistics and Bayesian computation. *J. R. Statist. Soc.* B, **55**, 25–37.

Besag, J., Green, P. J., Higdon, D. and Mengersen, K. (1995) Bayesian computation and stochastic systems. *Statist. Sci.*, **10**, 3–41.

Clayton, D. G. (1995) Generalized linear mixed models. In *Markov Chain Monte Carlo in Practice* (eds W. R. Gilks, S. Richardson and D. J. Spiegelhalter), pp. 275–301. London: Chapman & Hall.

DuMouchel, W. and Waternaux, C. (1992) Discussion on hierarchical models for combining information and for meta-analyses (by C. N. Morris. and L. Normand). In *Bayesian Statistics 4* (eds J. M. Bernardo, J. Berger, A. P. Dawid and A. F. M. Smith), pp. 338–339. Oxford: Oxford University Press.

Fosdick, L. D. (1963) Monte Carlo calculations on the Ising lattice. *Meth. Comput. Phys.*, **1**, 245–280.

Gelfand, A. E. and Sahu, S. K. (1994) On Markov chain Monte Carlo acceleration. *J. Comp. Graph. Statist.*, **3**, 261–267.

Gelfand, A. E., Sahu, S. K. and Carlin, B. P. (1995a) Efficient parametrizations for normal linear mixed models. *Biometrika*, (in press).

Gelfand, A. E., Sahu, S. K. and Carlin, B. P. (1995b) Efficient parametrizations for generalized linear mixed models. In *Bayesian Statistics 5* (eds J. M. Bernardo, J. Berger, A. P. Dawid and A. F. M. Smith). Oxford: Oxford University Press (in press).

Gelman, A., Roberts, G. O. and Gilks, W. R. (1995) Efficient Metropolis jumping rules. In *Bayesian Statistics 5* (eds J. M. Bernardo, J. Berger, A. P. Dawid and A. F. M. Smith). Oxford: Oxford University Press (in press).

Geweke, J. (1989) Bayesian inference in econometric models using Monte Carlo integration. *Econometrika*, **57**, 1317–1339.

Geyer, C. J. (1991) Markov chain Monte Carlo maximum likelihood. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface* (ed. E. M. Keramidas), pp. 156–163. Fairfax Station: Interface Foundation.

Geyer, C. J. (1993) Estimating normalising constants and reweighting mixtures in Markov chain Monte Carlo. *Technical Report 589*, School of Statistics, University of Minnesota.

Geyer, C. J. and Thompson, E. A. (1995) Annealing Markov chain Monte Carlo with applications to pedigree analysis. *J. Am. Statist. Ass.*, (in press).

Gilks, W. R. (1995a) Discussion on efficient parametrizations for generalized linear mixed models (by A. E. Gelfand, S. K. Sahu, and B. P. Carlin. In *Bayesian Statistics 5* (eds J. M. Bernardo, J. Berger, A. P. Dawid and A. F. M. Smith). Oxford: Oxford University Press (in press).

Gilks, W. R. (1995b) Full conditional distributions. In *Markov Chain Monte Carlo in Practice* (eds W. R. Gilks, S. Richardson and D. J. Spiegelhalter), pp. 75–88. London: Chapman & Hall.

Gilks, W. R., Best, N. G. and Tan, K. K. C. (1995a) Adaptive rejection Metropolis sampling within Gibbs sampling. *Appl. Statist.*, (in press).

Gilks, W. R., Richardson, S. and Spiegelhalter, D. J. (1995b) Introducing Markov chain Monte Carlo. In *Markov Chain Monte Carlo in Practice* (eds W. R. Gilks, S. Richardson and D. J. Spiegelhalter), pp. 1–19. London: Chapman & Hall.

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

Goodman, J. and Sokal, A. D. (1989) Multigrid Monte-Carlo method: conceptual foundations. *Phys. Rev. D*, **40**, 2035–2071.

Green, P. J. (1995) MCMC in image analysis. In *Markov Chain Monte Carlo in Practice* (eds W. R. Gilks, S. Richardson and D. J. Spiegelhalter), pp. 381–399. London: Chapman & Hall.

Grenander, U. and Miller, M. I. (1994) Representation of knowledge in complex systems. *J. R. Statist. Soc. B*, **56**, 549–603.

Hastings, W. K. (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 (eds J. M. Bernardo, J. Berger, A. P. Dawid and A. F. M. Smith), pp. 227–246. Oxford: Oxford University Press.

Hwang, C.-R., Hwang-Ma S.-Y. and Shen, S.-J. (1993). Accelerating Gaussian diffusions. *Ann. Appl. Prob.*, **3**, 897–913.

Jennison, C. (1993) Discussion on the meeting on the Gibbs sampler and other Markov chain Monte Carlo methods. *J. R. Statist. Soc.* B, **55**, 54–56.

Marinari, E. and Parisi, G. (1992). Simulated tempering: a new Monte Carlo scheme. *Europhys. Lett.*, **19**, 451–458.

Müller, P. (1994) A generic approach to posterior integration and Gibbs sampling. Technical report, Institute of Statistics and Decision Sciences, Duke University.

Mykland, P. Tierney, L. and Yu, B. (1995) Regeneration in Markov chain samplers. *J. Am. Statist. Ass.*, **90**, 233–241.

Neal, R. M. (1993) Probabilistic inference using Markov chain Monte Carlo methods. Technical report, Department of Computer Science, University of Toronto.

Peskun, P. H. (1973) Optimum Monte-Carlo sampling using Markov chains. *Biometrika*, **60**, 607–612.

Phillips, D. B. and Smith, A. F. M. (1995) Bayesian model comparison via jump diffusions. In *Markov Chain Monte Carlo in Practice* (eds W. R. Gilks, S. Richardson and D. J. Spiegelhalter), pp. 215–239. London: Chapman & Hall.

Ripley, B. D. (1987) *Stochastic Simulation*. New York: Wiley.

Roberts, G. O. (1995) Markov chain concepts related to sampling algorithmms. In *Markov Chain Monte Carlo in Practice* (eds W. R. Gilks, S. Richardson and D. J. Spiegelhalter), pp. 45–57. London: Chapman & Hall.

Roberts, G. O. and Gilks, W. R. (1994) Convergence of adaptive direction sampling. *J. Multiv. Anal.*, **49**, 287–298.

Roberts, G. O. and Tweedie, R. L. (1995) Stability of Langevin algorithms. *In preparation*.

Ross, G. S. J. (1990) *Nonlinear Estimation*. New York: Springer-Verlag.

Schmeiser, B. and Chen, M.-H. (1991) General hit-and-run Monte Carlo sampling for evaluating multidimensional integrals. Technical report, School of Industrial Engineering, Purdue University.

Sheehan, N. and Thomas, A. (1993) On the irreducibility of a Markov chain defined on a space of genotype configurations by a sampling scheme. *Biometrics*, **49**, 163–175.

Spiegelhalter, D. J., Thomas, A. and Best, N. G. (1995a). Computation on Bayesian graphical models. In *Bayesian Statistics 5* (eds J. M. Bernardo, J. Berger, A. P. Dawid and A. F. M. Smith). Oxford: Oxford University Press (in press).

Spiegelhalter, D. J., Best, N. G., Gilks, W. R. and Inskip, H. (1995b) Hepatitis B: a case study in MCMC methods. In *Markov Chain Monte Carlo in Practice* (eds W. R. Gilks, S. Richardson and D. J. Spiegelhalter), pp. 21–43. London: Chapman & Hall.

Swendsen, R. H. and Wang, J.-S. (1987) Nonuniversal critical dynamics in Monte carlo simulations. *Phs. Rev. Lett.*, **58**, 86–88.

Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *J. Am. Statist. Ass.*, **82**, 528–540.

Thomas, D. C. and Gauderman, W. J. (1995) Gibbs sampling methods in genetics. In *Markov Chain Monte Carlo in Practice* (eds W. R. Gilks, S. Richardson and D. J. Spiegelhalter), pp. 419–440. London: Chapman & Hall.

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

Tierney, L. (1995) Introduction to general state-space Markov chain theory. In *Markov Chain Monte Carlo in Practice* (eds W. R. Gilks, S. Richardson and D. J. Spiegelhalter), pp. 59–74. London: Chapman & Hall.

Vines, S. K. and Gilks, W. R. (1994) Reparameterising random interactions for Gibbs sampling. Technical report, MRC Biostatistics Unit, Cambridge.

Vines, S. K., Gilks, W. R. and Wild, P. (1995) Fitting multiple random effects models. Technical report, MRC Biostatistics Unit, Cambridge.

Wakefield, J. C. (1992) Discussion on parameterization issues in Bayesian inference (by S. E. Hills and A. F. M. Smith). In *Bayesian Statistics 4* (eds J. M. Bernardo, J. Berger, A. P. Dawid and A. F. M. Smith), pp. 243–244. Oxford: Oxford University Press.

7

Implementing MCMC

Adrian E Raftery

Steven M Lewis

7.1 Introduction

When implementing MCMC, it is important to determine how long the simulation should be run, and to discard a number of initial 'burn-in' iterations (see Gilks *et al.*, 1995: this volume). Saving all simulations from a MCMC run can consume a large amount of storage, especially when consecutive iterations are highly correlated, necessitating a long run. Therefore it is sometimes convenient to save only every k^{th} iteration ($k > 1$). This is sometimes referred to as *thinning* the chain. While neither burn-in nor thinning are mandatory practices, they both reduce the amount of data saved from a MCMC run.

In this chapter, we outline a way of determining in advance the number of iterations needed for a given level of precision in a MCMC algorithm. The method is introduced in Section 7.2, and in Section 7.3 we describe the `gibbsit` software which implements it and is available free of charge from StatLib. In Section 7.4, we show how the output from this method can also be used to diagnose lack of convergence or slow convergence due to bad starting values, high posterior correlations, or 'stickiness' (slow mixing) of the chain. In Section 7.5, we describe how the methods can be combined with ideas of Müller (1991) and Gelman *et al.* (1995) to yield an automatic generic Metropolis algorithm.

For simplicity, the discussion is in the context of a single long chain. However, as discussed in Section 7.6, the same basic ideas can also be used to determine the number of iterations and diagnose slow convergence when multiple sequences are used, as advocated by Gelman and Rubin (1992b): see also Gelman (1995: this volume).