Part I

Foundations, Methodology, and Algorithms

1

Introduction to Markov Chain Monte Carlo

Charles J. Geyer

1.1 History

Despite a few notable uses of simulation of random processes in the pre-computer era (Hammersley and Handscomb, 1964, Section 1.2; Stigler, 2002, Chapter 7), practical widespread use of simulation had to await the invention of computers. Almost as soon as computers were invented, they were used for simulation (Hammersley and Handscomb, 1964, Section 1.2). The name "Monte Carlo" started as cuteness—gambling was then (around 1950) illegal in most places, and the casino at Monte Carlo was the most famous in the world—but it soon became a colorless technical term for simulation of random processes.

Markov chain Monte Carlo (MCMC) was invented soon after ordinary Monte Carlo at Los Alamos, one of the few places where computers were available at the time. Metropolis et al. (1953)* simulated a liquid in equilibrium with its gas phase. The obvious way to find out about the thermodynamic equilibrium is to simulate the dynamics of the system, and let it run until it reaches equilibrium. The *tour de force* was their realization that they did not need to simulate the exact dynamics; they only needed to simulate some Markov chain having the same equilibrium distribution. Simulations following the scheme of Metropolis et al. (1953) are said to use the *Metropolis algorithm*. As computers became more widely available, the Metropolis algorithm was widely used by chemists and physicists, but it did not become widely known among statisticians until after 1990. Hastings (1970) generalized the Metropolis algorithm, and simulations following his scheme are said to use the *Metropolis–Hastings* algorithm. A special case of the Metropolis–Hastings algorithm was introduced by Geman and Geman (1984), apparently without knowledge of earlier work. Simulations following their scheme are said to use the *Gibbs sampler*. Much of Geman and Geman (1984) discusses optimization to find the posterior mode rather than simulation, and it took some time for it to be understood in the spatial statistics community that the Gibbs sampler simulated the posterior distribution, thus enabling full Bayesian inference of all kinds. A methodology that was later seen to be very similar to the Gibbs sampler was introduced by Tanner and Wong (1987), again apparently without knowledge of earlier work. To this day, some refer to the Gibbs sampler as "data augmentation" following these authors. Gelfand and Smith (1990) made the wider Bayesian community aware of the Gibbs sampler, which up to that time had been known only in the spatial statistics community. Then it took off; as of this writing, a search for Gelfand and Smith (1990) on Google Scholar yields 4003 links to other works. It was rapidly realized that most Bayesian inference could

* The fifth author was Edward Teller, the "father of the hydrogen bomb."

be done by MCMC, whereas very little could be done without MCMC. It took a while for researchers to properly understand the theory of MCMC (Geyer, 1992; Tierney, 1994) and that all of the aforementioned work was a special case of the notion of MCMC. Green (1995) generalized the Metropolis–Hastings algorithm, as much as it can be generalized. Although this terminology is not widely used, we say that simulations following his scheme use the *Metropolis–Hastings–Green* algorithm. MCMC is not used only for Bayesian inference. Likelihood inference in cases where the likelihood cannot be calculated explicitly due to missing data or complex dependence can also use MCMC (Geyer, 1994, 1999; Geyer and Thompson, 1992, 1995, and references cited therein).

1.2 Markov Chains

A sequence $X_1, X_2, \ldots$ of random elements of some set is a *Markov chain* if the conditional distribution of X_{n+1} given $X_1, \ldots, X_n$ depends on X_n only. The set in which the X_i take values is called the *state space* of the Markov chain.

A Markov chain has *stationary transition probabilities* if the conditional distribution of X_{n+1} given X_n does not depend on n. This is the main kind of Markov chain of interest in MCMC. Some kinds of adaptive MCMC (Chapter 4, this volume) have nonstationary transition probabilities. In this chapter we always assume stationary transition probabilities.

The joint distribution of a Markov chain is determined by

- The marginal distribution of X_1, called the *initial distribution*
- The conditional distribution of X_{n+1} given X_n, called the *transition probability distribution* (because of the assumption of stationary transition probabilities, this does not depend on n)

People introduced to Markov chains through a typical course on stochastic processes have usually only seen examples where the state space is finite or countable. If the state space is finite, written $\{x_1, \ldots, x_n\}$, then the initial distribution can be associated with a vector $\lambda = (\lambda_1, \ldots, \lambda_n)$ defined by

$$\Pr(X_1 = x_i) = \lambda_i, \quad i = 1, \ldots, n,$$

and the transition probabilities can be associated with a matrix P having elements p_{ij} defined by

$$\Pr(X_{n+1} = x_j \mid X_n = x_i) = p_{ij}, \quad i = 1, \ldots, n \quad \text{and} \quad j = 1, \ldots, n.$$

When the state space is countably infinite, we can think of an infinite vector and matrix. But most Markov chains of interest in MCMC have uncountable state space, and then we cannot think of the initial distribution as a vector or the transition probability distribution as a matrix. We must think of them as an unconditional probability distribution and a conditional probability distribution.

1.3 Computer Programs and Markov Chains

Suppose you have a computer program

```
Initialize x
repeat {
    Generate pseudorandom change to x
    Output x
}
```

If x is the entire state of the computer program exclusive of random number generator seeds (which we ignore, pretending pseudorandom is random), this is MCMC. It is important that x must be the entire state of the program. Otherwise the resulting stochastic process need not be Markov.

There is not much structure here. Most simulations can be fit into this format. Thus most simulations can be thought of as MCMC if the entire state of the computer program is considered the state of the Markov chain. Hence, MCMC is a very general simulation methodology.

1.4 Stationarity

A sequence $X_1, X_2, \ldots$ of random elements of some set is called a *stochastic process* (Markov chains are a special case). A stochastic process is *stationary* if for every positive integer k the distribution of the k-tuple

$$(X_{n+1}, \ldots, X_{n+k})$$

does not depend on n. A Markov chain is stationary if it is a stationary stochastic process. In a Markov chain, the conditional distribution of $(X_{n+2}, \ldots, X_{n+k})$ given X_{n+1} does not depend on n. It follows that a Markov chain is stationary if and only if the marginal distribution of X_n does not depend on n.

An initial distribution is said to be *stationary* or *invariant* or *equilibrium* for some transition probability distribution if the Markov chain specified by this initial distribution and transition probability distribution is stationary. We also indicate this by saying that the transition probability distribution *preserves* the initial distribution.

Stationarity implies stationary transition probabilities, but not vice versa. Consider an initial distribution concentrated at one point. The Markov chain can be stationary if and only if all iterates are concentrated at the same point, that is, $X_1 = X_2 = \ldots$, so the chain goes nowhere and does nothing. Conversely, any transition probability distribution can be combined with any initial distribution, including those concentrated at one point. Such a chain is usually not stationary (even though the transition probabilities are stationary).

Having an equilibrium distribution is an important property of a Markov chain transition probability. In Section 1.8 below, we shall see that MCMC samples the equilibrium distribution, whether the chain is stationary or not. Not all Markov chains have equilibrium distributions, but all Markov chains used in MCMC do. The Metropolis–Hastings–Green (MHG) algorithm (Sections 1.12.2, 1.17.3.2, and 1.17.4.1 below) constructs transition probability mechanisms that preserve a specified equilibrium distribution.

1.5 Reversibility

A transition probability distribution is *reversible* with respect to an initial distribution if, for the Markov chain $X_1, X_2, \ldots$ they specify, the distribution of pairs (X_i, X_{i+1}) is exchangeable.

A Markov chain is *reversible* if its transition probability is reversible with respect to its initial distribution. Reversibility implies stationarity, but not vice versa. A reversible Markov chain has the same laws running forward or backward in time, that is, for any i and k the distributions of $(X_{i+1}, \ldots, X_{i+k})$ and $(X_{i+k}, \ldots, X_{i+1})$ are the same. Hence the name.

Reversibility plays two roles in Markov chain theory. All known methods for constructing transition probability mechanisms that preserve a specified equilibrium distribution in non-toy problems are special cases of the MHG algorithm, and all of the elementary updates constructed by the MHG algorithm are reversible (which accounts for its other name, the "reversible jump" algorithm). Combining elementary updates by composition (Section 1.12.7 below) may produce a combined update mechanism that is not reversible, but this does not diminish the key role played by reversibility in constructing transition probability mechanisms for MCMC. The other role of reversibility is to simplify the Markov chain central limit theorem (CLT) and asymptotic variance estimation. In the presence of reversibility the Markov chain CLT (Kipnis and Varadhan, 1986; Roberts and Rosenthal, 1997) is much sharper and the conditions are much simpler than without reversibility. Some methods of asymptotic variance estimation (Section 1.10.2 below) only work for reversible Markov chains but are much simpler and more reliable than analogous methods for nonreversible chains.

1.6 Functionals

If $X_1, X_2, \ldots$ is a stochastic process and g is a real-valued function on its state space, then the stochastic process $g(X_1), g(X_2), \ldots$ having state space $\mathbb{R}$ is said to be a *functional* of $X_1, X_2, \ldots$.

If $X_1, X_2, \ldots$ is a Markov chain, then a functional $g(X_1), g(X_2), \ldots$ is usually not a Markov chain. The conditional distribution of X_{n+1} given $X_1, \ldots, X_n$ depends only on X_n, but this does not, in general, imply that the conditional distribution of $g(X_{n+1})$ given $g(X_1), \ldots, g(X_n)$ depends only on $g(X_n)$. Nevertheless, functionals of Markov chains have important properties not shared by other stochastic processes.

1.7 The Theory of Ordinary Monte Carlo

Ordinary Monte Carlo (OMC), also called "independent and identically distributed (i.i.d.) Monte Carlo" or "good old-fashioned Monte Carlo," is the special case of MCMC in which $X_1, X_2, \ldots$ are independent and identically distributed, in which case the Markov chain is stationary and reversible.

Suppose you wish to calculate an expectation

$$\mu = \mathrm{E}\{g(X)\}, \tag{1.1}$$

where g is a real-valued function on the state space, but you cannot do it by exact methods (integration or summation using pencil and paper, a computer algebra system, or exact numerical methods). Suppose you can simulate $X_1, X_2, \ldots$ i.i.d. having the same distribution as X. Define

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} g(X_i). \tag{1.2}$$

If we introduce the notation $Y_i = g(X_i)$, then the Y_i are i.i.d. with mean μ and variance

$$\sigma^2 = \operatorname{var}\{g(X)\}, \tag{1.3}$$

$\hat{\mu}_n$ is the sample mean of the Y_i, and the CLT says that

$$\hat{\mu}_n \approx N\left(\mu, \frac{\sigma^2}{n}\right). \tag{1.4}$$

The variance in the CLT can be estimated by

$$\hat{\sigma}_n^2 = \frac{1}{n} \sum_{i=1}^{n} \bigl(g(X_i) - \hat{\mu}_n\bigr)^2, \tag{1.5}$$

which is the empirical variance of the Y_i. Using the terminology of Section 1.6, we can also say that $\hat{\mu}_n$ is the sample mean of the functional $g(X_1), g(X_2), \ldots$ of $X_1, X_2, \ldots$.

The theory of OMC is just elementary statistics. For example, $\hat{\mu}_n \pm 1.96 \cdot \hat{\sigma}_n/\sqrt{n}$ is an asymptotic 95% confidence interval for μ. Note that OMC obeys what an elementary statistics text (Freedman et al., 2007) calls the *square root law:* statistical accuracy is inversely proportional to the square root of the sample size. Consequently, the accuracy of Monte Carlo methods is limited. Each additional significant figure, a tenfold increase in accuracy, requires a hundredfold increase in the sample size.

The only tricky issue is that the randomness involved is the pseudorandomness of computer simulation, rather than randomness of real-world phenomena. Thus it is a good idea to use terminology that emphasizes the difference. We call Equation 1.2 the *Monte Carlo approximation* or *Monte Carlo calculation* of μ, rather than the "point estimate" or "point estimator" of μ, as we would if not doing Monte Carlo. We call n the *Monte Carlo sample size*, rather than just the "sample size." We call $\hat{\sigma}_n/\sqrt{n}$ the *Monte Carlo standard error* (MCSE), rather than just the "standard error." We also do not refer to Equation 1.1 as an unknown parameter, even though we do not know its value. It is simply the expectation we are trying to calculate, known in principle, although unknown in practice, since we do not know how to calculate it other than by Monte Carlo approximation.

It is especially important to use this terminology when applying Monte Carlo to statistics. When the expectation (Equation 1.1) arises in a statistical application, there may already be a sample size in this application, which is unrelated to the Monte Carlo sample size, and there may already be standard errors unrelated to MCSEs. It can be hopelessly confusing if these are not carefully distinguished.

1.8 The Theory of MCMC

The theory of MCMC is just like the theory of OMC, except that stochastic dependence in the Markov chain changes the standard error. We start as in OMC with an expectation (Equation 1.1) that we cannot do other than by Monte Carlo. To begin the discussion, suppose that $X_1, X_2, \ldots$ is a stationary Markov chain having initial distribution the same as the distribution of X. We assume that the Markov chain CLT (Equation 1.4) holds, where now

$$\sigma^2 = \operatorname{var}\{g(X_i)\} + 2\sum_{k=1}^{\infty} \operatorname{cov}\{g(X_i), g(X_{i+k})\} \tag{1.6}$$

(this formula is correct only for stationary Markov chains; see below for nonstationary chains). Since the asymptotic variance (Equation 1.6) is more complicated than the i.i.d. case (Equation 1.3), it cannot be estimated by Equation 1.5. It can, however, be estimated in several ways discussed below (Section 1.10). Conditions for the Markov chain CLT to hold (Chan and Geyer, 1994; Jones, 2004; Roberts and Rosenthal, 1997, 2004; Tierney, 1994) are beyond the scope of this chapter.

Now we come to a somewhat confusing issue. We never use stationary Markov chains in MCMC, because if we could simulate X_1 so that it has the invariant distribution, then we could also simulate $X_2, X_3, \ldots$ in the same way and do OMC. It is a theorem, however, that, under a condition (Harris recurrence) that is easier to verify than the CLT (Chan and Geyer, 1994; Tierney, 1994), if the CLT holds for one initial distribution and transition probability, then it holds for all initial distributions and that same transition probability (Meyn and Tweedie, 1993, Proposition 17.1.6), and the asymptotic variance is the same for all initial distributions. Although the theoretical asymptotic variance formula (Equation 1.6) contains variances and covariances for the stationary Markov chain, it also gives the asymptotic variance for nonstationary Markov chains having the same transition probability distribution (but different initial distributions). In practice, this does not matter, because we can never calculate (Equation 1.6) exactly except in toy problems and must estimate it from our simulations.

1.8.1 Multivariate Theory

Suppose that we wish to approximate by Monte Carlo (Equation 1.1) where we change notation so that μ is a vector with components μ_r and $g(x)$ is a vector with components $g_r(x)$. Our Monte Carlo estimator is still given by Equation 1.2, which is now also a vector equation because each $g(X_i)$ is a vector. Then the multivariate Markov chain CLT says that

$$\hat{\mu}_n \approx N(\mu, n^{-1}\Sigma),$$

where

$$\Sigma = \operatorname{var}\{g(X_i)\} + 2\sum_{k=1}^{\infty} \operatorname{cov}\{g(X_i), g(X_{i+k})\}, \tag{1.7}$$

and where, although the right-hand sides of Equations 1.6 and 1.7 are the same, they mean different things: in Equation 1.7 $\operatorname{var}\{g(X_i)\}$ denotes the matrix with components $\operatorname{cov}\{g_r(X_i), g_s(X_i)\}$ and $\operatorname{cov}\{g(X_i), g(X_{i+k})\}$ denotes the matrix with components $\operatorname{cov}\{g_r(X_i), g_s(X_{i+k})\}$.

Conditions for the multivariate CLT to hold are essentially the same as for the univariate CLT. By the Cramér–Wold theorem, the multivariate convergence in distribution $Z_n \xrightarrow{\mathcal{D}} Z$ holds if and only if the univariate convergence in distribution $t'Z_n \xrightarrow{\mathcal{D}} t'Z$ holds for every nonrandom vector t. Thus the multivariate CLT essentially follows from the univariate CLT, and is often not discussed. It is important, however, for users to understand that the multivariate CLT does hold and can be used when needed.

1.8.2 The Autocovariance Function

We introduce terminology for the covariances that appear in Equation 1.6:

$$\gamma_k = \operatorname{cov}\{g(X_i), g(X_{i+k})\} \tag{1.8}$$

is called the *lag-k autocovariance* of the functional $g(X_1), g(X_2), \ldots$. Recall that in Equation 1.8 as in Equation 1.6 the covariances refer to the stationary chain with the same transition probability distribution as the chain being used. The variance that appears in Equation 1.6 is then γ_0. Hence, (Equation 1.6) can be rewritten

$$\sigma^2 = \gamma_0 + 2\sum_{k=1}^{\infty} \gamma_k. \tag{1.9}$$

The function $k \mapsto \gamma_k$ is called the *autocovariance function* of the functional $g(X_1), g(X_2), \ldots$, and the function $k \mapsto \gamma_k/\gamma_0$ is called the *autocorrelation function* of this functional.

The natural estimator of the autocovariance function is

$$\hat{\gamma}_k = \frac{1}{n}\sum_{i=1}^{n-k}[g(X_i) - \hat{\mu}_n][g(X_{i+k}) - \hat{\mu}_n] \tag{1.10}$$

It might be thought that one should divide by $n - k$ instead of n, but the large k terms are already very noisy so dividing by $n - k$ only makes a bad situation worse. The function $k \mapsto \hat{\gamma}_k$ is called the *empirical autocovariance function* of the functional $g(X_1), g(X_2), \ldots$, and the function $k \mapsto \hat{\gamma}_k/\hat{\gamma}_0$ is called the *empirical autocorrelation function* of this functional.

1.9 AR(1) Example

We now look at a toy problem for which exact calculation is possible. An AR(1) process (AR stands for autoregressive) is defined recursively by

$$X_{n+1} = \rho X_n + Y_n, \tag{1.11}$$

where Y_n are i.i.d. $N(0, \tau^2)$ and X_1 may have any distribution with finite variance. From Equation 1.11 we get

$$\operatorname{cov}(X_{n+k}, X_n) = \rho \operatorname{cov}(X_{n+k-1}, X_n) = \ldots = \rho^{k-1} \operatorname{cov}(X_{n-1}, X_n) = \rho^k \operatorname{var}(X_n). \tag{1.12}$$

If the process is stationary, then

$$\operatorname{var}(X_n) = \operatorname{var}(X_{n+1}) = \rho^2 \operatorname{var}(X_n) + \operatorname{var}(Y_n)$$

so

$$\operatorname{var}(X_n) = \frac{\tau^2}{1 - \rho^2} \tag{1.13}$$

and since variances are nonnegative, we must have $\rho^2 < 1$. Since a linear combination of independent normal random variables is normal, we see that the normal distribution with mean zero and variance (Equation 1.13) is invariant. Define υ^2 to be another notation for the right-hand side of Equation 1.13 so the invariant distribution is $N(0, \upsilon^2)$.

It can be shown that this is the unique invariant distribution and this Markov chain obeys the CLT. The variance in the CLT is

$$\begin{aligned}
\sigma^2 &= \operatorname{var}(X_i) + 2 \sum_{k=1}^{\infty} \operatorname{cov}(X_i, X_{i+k}) \\
&= \frac{\tau^2}{1 - \rho^2} \left(1 + 2 \sum_{k=1}^{\infty} \rho^k \right) \\
&= \frac{\tau^2}{1 - \rho^2} \left(1 + \frac{2\rho}{1 - \rho} \right) \\
&= \frac{\tau^2}{1 - \rho^2} \cdot \frac{1 + \rho}{1 - \rho} \\
&= \upsilon^2 \cdot \frac{1 + \rho}{1 - \rho}.
\end{aligned} \tag{1.14}$$

1.9.1 A Digression on Toy Problems

It is hard to know what lessons to learn from a toy problem. Unless great care is taken to point out which features of the toy problem are like real applications and which unlike, readers may draw conclusions that do not apply to real-world problems.

Here we are supposed to pretend that we do not know the invariant distribution, and hence we do not know that the expectation we are trying to estimate, $\mu = \mathrm{E}(X)$, where X has the invariant distribution, is zero.

We cannot be interested in any functional of the Markov chain other than the one induced by the identity function, because we cannot do the analog of Equation 1.14 for any function g other than the identity function, and thus would not have a closed-form expression for the variance in the Markov chain CLT, which is the whole point of this toy problem.

Observe that Equation 1.14 goes to infinity as $\rho \to 1$. Thus in order to obtain a specified accuracy for $\hat{\mu}_n$ as an approximation to μ, say $\sigma/\sqrt{n} = \varepsilon$, we may need a very large Monte Carlo sample size n. How large n must be depends on how close ρ is to one. When we pretend that we do not know the asymptotic variance (Equation 1.14), which we should do because the asymptotic variance is never known in real applications, all we can conclude is that we may need the Monte Carlo sample size to be very large and have no idea how large.

We reach the same conclusion if we are only interested in approximation error relative to the standard deviation υ of the invariant distribution, because

$$\frac{\sigma^2}{\upsilon^2} = \frac{1+\rho}{1-\rho} \tag{1.15}$$

also goes to infinity as $\rho \to 1$.

1.9.2 Supporting Technical Report

In order to avoid including laborious details of examples while still making all examples fully reproducible, those details are relegated to a technical report (Geyer, 2010a) or the vignettes for the R package `mcmc` (Geyer, 2010b). All calculations in this technical report or those package vignettes are done using the R function `Sweave`, so all results in them are actually produced by the code shown therein and hence are fully reproducible by anyone who has R. Moreover, anyone can download the `Sweave` source for the technical report from the URL given in the references at the end of this chapter or find the `Sweave` source for the package vignettes in the `doc` directory of any installation of the `mcmc` package, separate the R from the LaTeX using the `Stangle` function, and play with it to see how the examples work.

1.9.3 The Example

For our example, we choose $\rho = 0.99$ and Monte Carlo sample size $n = 10^4$. This makes the MCSE about 14% of the standard deviation of the invariant distribution, which is a pretty sloppy approximation. To get the relative MCSE down to 10%, we would need $n = 2 \times 10^4$. To get the relative MCSE down to 1%, we would need $n = 2 \times 10^6$.

Figure 1.1 shows a time series plot of one MCMC run for this AR(1) process. From this plot we can see that the series seems stationary—there is no obvious trend or change in spread. We can also get a rough idea of how much dependence there is in the chain by

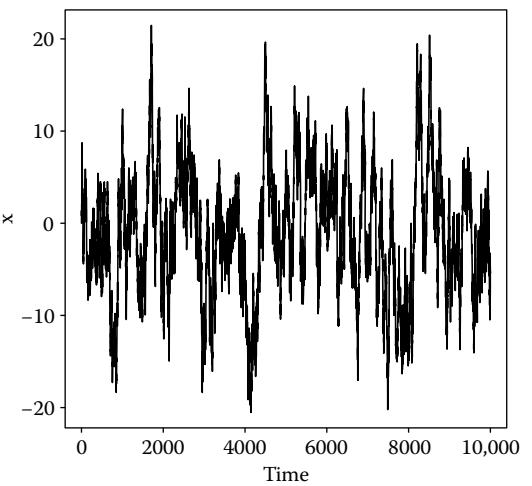

FIGURE 1.1
Time series plot for AR(1) example.

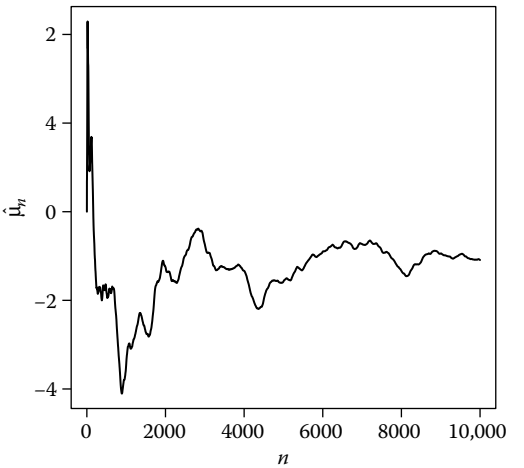

FIGURE 1.2
Running averages plot for AR(1) example.

counting large wiggles. The ratio of the variance in the CLT to the variance of the invariant distribution (Equation 1.15) is 199 for this example. Hence, this MCMC sample is about as useful as an i.i.d. sample with the same marginal distribution of sample size $10^4/199 \approx 50$.

Figure 1.2 shows a running averages plot for the same run shown in Figure 1.1. For some reason, these running averages plots seem popular among MCMC users although they provide no useful information. We know that MCMC, like OMC, obeys the square root law. A plot like Figure 1.2 does illustrate that $1/\sqrt{n}$ is a decreasing function of n, but not much else. Elementary statistics texts (Freedman et al., 2007, p. 276) often include one (and only one) figure like our Figure 1.2 to illustrate to naive students how the law of averages works. We have included Figure 1.2 only as an example of what not to do. In particular, such running averages plots should never be used to illustrate talks, since they tell the audience nothing they do not already know. Show a time series plot, like Figure 1.1, instead.

Figure 1.3 shows an autocorrelation plot for the same run shown in Figure 1.1. The black bars show the empirical autocorrelation function (ACF) defined in Section 1.8.2. We could let the domain of the ACF be zero to $n - 1$, but the R function `acf` cuts the plot at the argument `lag.max`. The `acf` function automatically adds the horizontal dashed lines, which the documentation for `plot.acf` says are 95% confidence intervals assuming white noise input. The dotted curve is the simulation truth autocorrelation function ρ^k derived from Equation 1.12. In the spirit of this toy problem, we are supposed to pretend we do not know the dotted curve, since we would not have its analog in any real application. We can see, however, how well (not very) the empirical ACF matches the theoretical ACF.

It should come as no surprise that the empirical ACF estimates the theoretical ACF less well than $\hat{\mu}_n$ estimates μ. Even in i.i.d. sampling, the mean is always much better estimated than the variance.

The ACF is well enough estimated, however, to give some idea how far significant autocorrelation extends in our Markov chain. Of course, the theoretical autocorrelation is nonzero for all lags, no matter how large, but we know (although we pretend we do not) that they decrease exponentially fast. They are not practically significantly different from zero past lag 500.

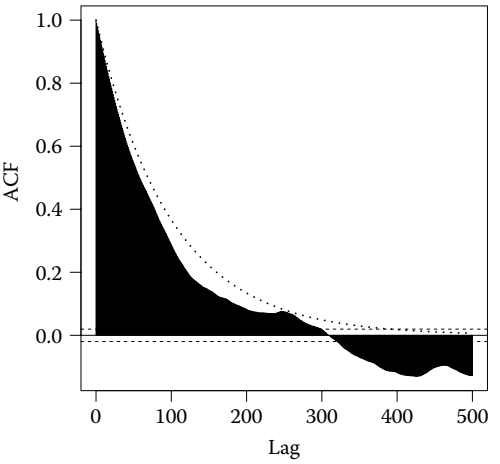

FIGURE 1.3
Autocorrelation plot for AR(1) Example. Dashed lines: 95% confidence intervals assuming white noise input. Dotted curve: simulation truth autocorrelation function.

1.10 Variance Estimation

Many methods of variance estimation have been proposed. Most come from the time series literature and are applicable to arbitrary stationary stochastic processes, not just to Markov chains. We will cover only a few very simple, but very effective, methods.

1.10.1 Nonoverlapping Batch Means

A *batch* is simply a subsequence of consecutive iterates of the Markov chain $X_{k+1}, \ldots, X_{k+b}$. The number b is called the *batch length*. If we assume the Markov chain is stationary, then all batches of the same length have the same joint distribution, and the CLT applies to each batch. The batch mean

$$\frac{1}{b} \sum_{j=1}^{b} g(X_{k+j})$$

is a Monte Carlo approximation of the expectation (Equation 1.1) we are trying to calculate, and its distribution is approximately $N(\mu, \sigma^2/b)$, where, as before, σ^2 is given by Equation 1.6. A batch of length b is just like the entire run of length n, except for length. The sample mean of a batch of length b is just like the sample mean of the entire run of length n, except that the asymptotic variance is σ^2/b instead of σ^2/n.

Suppose b divides n evenly. Divide the whole run into m nonoverlapping batches of length b. Average these batches:

$$\hat{\mu}_{b,k} = \frac{1}{b} \sum_{i=b(k-1)+1}^{bk} g(X_i). \tag{1.16}$$

Then

$$\frac{1}{m}\sum_{k=1}^{m}(\hat{\mu}_{b,k} - \hat{\mu}_n)^2 \tag{1.17}$$

estimates σ^2/b.

It is important to understand that the stochastic process $\hat{\mu}_{b,1}, \hat{\mu}_{b,2}, \ldots$ is also a functional of a Markov chain, not the original Markov chain but a different one. If S is the state space of the original Markov chain $X_1, X_2, \ldots$, then the batches

$$(X_{b(k-1)+1}, \ldots, X_{kb}), \quad k = 1, 2, \ldots$$

also form a Markov chain with state space S^b, because the conditional distribution of one batch $(X_{b(k-1)+1}, \ldots, X_{kb})$ given the past history actually depends only on $X_{b(k-1)}$, which is a component of the immediately preceding batch. The batch means are a functional of this Markov chain of batches.

Figure 1.4 shows a batch mean plot for the same run shown in Figure 1.1. The batch length is 500, the run length is 10^4, so the number of batches is 20. Like the running averages plot (Figure 1.2), we do not recommend this kind of plot for general use, because it does not show anything a sophisticated MCMC user should not already know. It is useful to show such a plot (once) in a class introducing MCMC, to illustrate the point that the stochastic process shown is a functional of a Markov chain. It is not useful for talks about MCMC.

Figure 1.5 shows the autocorrelation plot of the batch mean stochastic process for the same run shown in Figure 1.1, which shows the batches are not significantly correlated, because all of the bars except the one for lag 0 are inside the dashed lines. In this case, a confidence interval for the unknown expectation (Equation 1.1) is easily done using the R function `t.test`:

```
> t.test(batch)
        One Sample t-test

data:  batch
t = -1.177, df = 19, p-value = 0.2537
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -2.5184770  0.7054673
sample estimates:
 mean of x
-0.9065049
```

Here, `batch` is the vector of batch means which is plotted in Figure 1.4.

If this plot had shown the batches to be significantly correlated, then the method of batch means should not have been used because it would have a significant downward bias. However, the time series of batches can still be used, as explained in Section 1.10.2 below.

How does one choose the batch length? The method of batch means will work well only if the batch length b is large enough so that the infinite sum in Equation 1.9 is well approximated by the partial sum of the first b terms. Hence, when the method of batch means is used blindly with no knowledge of the ACF, b should be as large as possible. The only restriction on the length of batches is that the number of batches should be enough to

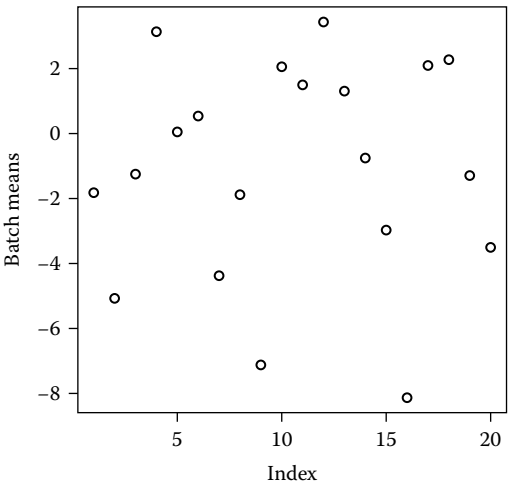

FIGURE 1.4
Batch mean plot for AR(1) example. Batch length 500.

get a reasonable estimate of variance. If one uses a t test, as shown above, then the t critical value corrects for the number of batches being small (Geyer, 1992; Schmeiser, 1982), but there is no point in the number of batches being so small that that the variance estimate is extremely unstable: 20–30 batches is a reasonable recommendation. One sometimes sees assumptions that the number of batches "goes to infinity" in theorems, but this is not necessary for simple MCSE calculation (Geyer, 1992, Section 3.2). If one is using estimated variance in a sequential stopping rule (Glynn and Whitt, 1991, 1992), then one does need the number of batches to go to infinity.

Meketon and Schmeiser (1984) pointed out that the batch means estimator of variance (Equation 1.17) is still valid if the batches are allowed to overlap, and a slight gain in efficiency is thereby achieved. For reasons explained in the following section, we do not

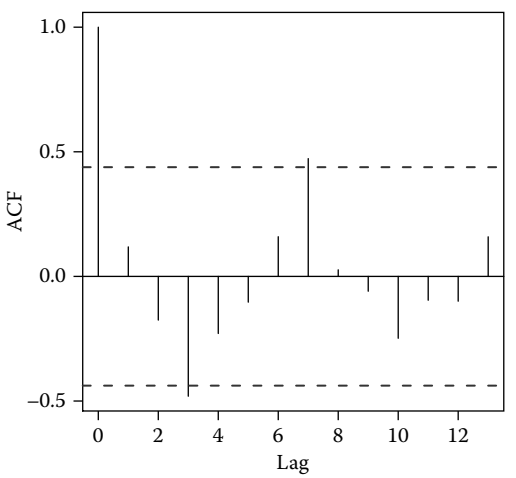

FIGURE 1.5
Autocorrelation plot of batch means for AR(1) example. Batch length 500.

recommend overlapping batch means, not because there is anything wrong with it, but because it does not fit together well with other methods we recommend.

1.10.2 Initial Sequence Methods

Another approach to variance estimation is to work directly with the representation (Equation 1.9) of the asymptotic variance. One cannot simply plug the empirical estimates (Equation 1.10) into Equation 1.9 because the variance of the high-lag terms does not decrease with lag, so as n goes to infinity an infinite amount of noise swamps the finite signal. Many solutions for this problem have been proposed in the time series literature (Geyer, 1992, Section 3.1 and references cited therein). But reversible Markov chains permit much simpler methods. Define

$$\Gamma_k = \gamma_{2k} + \gamma_{2k+1}. \tag{1.18}$$

Geyer (1992, Theorem 3.1) showed that the function $k \mapsto \Gamma_k$ is strictly positive, strictly decreasing, and strictly convex, and proposed three estimators of the asymptotic variance (Equation 1.9) that use these three properties, called the *initial positive sequence*, *initial monotone sequence*, and *initial convex sequence* estimators. Each is a consistent overestimate of the asymptotic variance (meaning the probability of underestimation by any fixed amount goes to zero as the Monte Carlo sample size goes to infinity) under no regularity conditions whatsoever (Geyer, 1992, Theorem 3.2). The initial convex sequence estimator is the best, because the smallest and still an asymptotic overestimate, but is a bit difficult to calculate. Fortunately, the R contributed package `mcmc` now has a function `initseq` that calculates all three estimators. We will only discuss the last. It forms

$$\widehat{\Gamma}_k = \hat{\gamma}_{2k} + \hat{\gamma}_{2k+1},$$

where $\hat{\gamma}_k$ is given by Equation 1.10, then finds the largest index m such that

$$\widehat{\Gamma}_k > 0, \quad k = 0, \ldots, m,$$

then defines $\widehat{\Gamma}_{m+1} = 0$, and then defines $k \mapsto \widetilde{\Gamma}_k$ to be the greatest convex minorant of $k \mapsto \widehat{\Gamma}_k$ over the range $0, \ldots, m+1$. Finally, it estimates

$$\hat{\sigma}^2_{\text{conv}} = -\hat{\gamma}_0 + 2 \sum_{k=0}^{m} \widetilde{\Gamma}_k. \tag{1.19}$$

Figure 1.6 shows a plot of the function $k \mapsto \widetilde{\Gamma}_k$ for the same run shown in Figure 1.1 compared to its theoretical value. When comparing this plot to Figure 1.3, remember that each index value in Figure 1.6 corresponds to two index values in Figure 1.3 because of the way Equation 1.18 is defined. Thus Figure 1.6 indicates significant autocorrelation out to about lag 300 (not 150).

The estimator of asymptotic variance (Equation 1.19) is calculated very simply in R:

```
> initseq(out)$var.con
[1] 7467.781
```

assuming the `mcmc` contributed package has already been loaded and `out` is the functional of the Markov chain for which the variance estimate is desired.

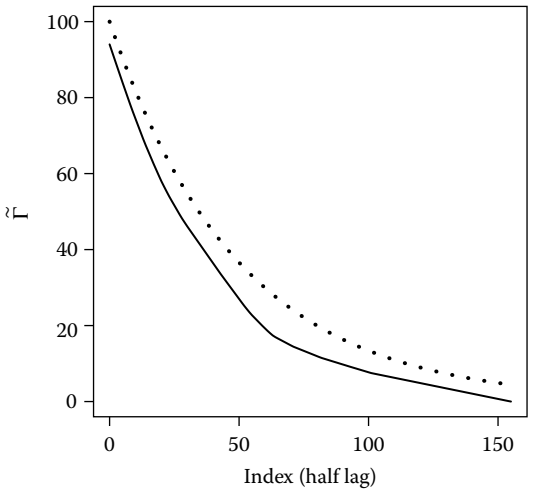

FIGURE 1.6
Plot of $\widetilde{\Gamma}$ for AR(1) example. Solid line: initial convex sequence estimator of Equation 1.18. Dotted line: theoretical value.

1.10.3 Initial Sequence Methods and Batch Means

When the original Markov chain is reversible, so is the chain of batches. Hence, initial sequence methods can be applied to a sequence of nonoverlapping batch means derived from a reversible Markov chain.

This means that the method of nonoverlapping batch means can be used without testing whether the batches are large enough. Simply process them with an initial sequence method, and the result is valid regardless of the batch length.

Here is how that works. Suppose we use a batch length of 50, which is too short.

```
> blen * var(batch)
[1] 2028.515
> blen * initseq(batch)$var.con
[1] 7575.506
```

The naive batch means estimator is terrible, less than a third of the size of the initial convex sequence estimator applied to the batch means (7575.506), but this is about the same as the initial convex sequence estimator applied to the original output (7467.781). So nothing is lost when only nonoverlapping batch means are output, regardless of the batch length used.

Partly for this reason, and partly because nonoverlapping batch means are useful for reducing the size of the output, whereas overlapping batch means are not, we do not recommend overlapping batch means and will henceforth always use the term *batch means* to mean nonoverlapping batch means.

1.11 The Practice of MCMC

The practice of MCMC is simple. Set up a Markov chain having the required invariant distribution, and run it on a computer. The folklore of simulation makes this seem more

complicated than it really is. None of this folklore is justified by theory and none of it actually helps users do good simulations, but, like other kinds of folklore, it persists despite its lack of validity.

1.11.1 Black Box MCMC

There is a great deal of theory about convergence of Markov chains. Unfortunately, none of it can be applied to get useful convergence information for most MCMC applications. Thus most users find themselves in the following situation we call *black box MCMC*:

1. You have a Markov chain having the required invariant distribution.
2. You know nothing other than that. The Markov chain is a "black box" that you cannot see inside. When run, it produces output. That is all you know. You know nothing about the transition probabilities of the Markov chain, nor anything else about its dynamics.
3. You know nothing about the invariant distribution except what you may learn from running the Markov chain.

Point 2 may seem extreme. You may know a lot about the particular Markov chain being used—for example, you may know that it is a Gibbs sampler—but if whatever you know is of no help in determining any convergence information about the Markov chain, then whatever knowledge you have is useless. Point 3 may seem extreme. Many examples in the MCMC literature use small problems that can be done by OMC or even by pencil and paper and for which a lot of information about the invariant distribution is available, but in complicated applications point 3 is often simply true.

1.11.2 Pseudo-Convergence

A Markov chain can appear to have converged to its equilibrium distribution when it has not. This happens when parts of the state space are poorly connected by the Markov chain dynamics: it takes many iterations to get from one part to another. When the time it takes to transition between these parts is much longer than the length of simulated Markov chain, then the Markov chain can appear to have converged but the distribution it appears to have converged to is the equilibrium distribution conditioned on the part in which the chain was started. We call this phenomenon *pseudo-convergence*.

This phenomenon has also been called "multimodality" since it may occur when the equilibrium distribution is multimodal. But multimodality does not cause pseudo-convergence when the troughs between modes are not severe. Nor does pseudo-convergence only happen when there is multimodality. Some of the most alarming cases of pseudo-convergence occur when the state space of the Markov chain is discrete and "modes" are not well defined (Geyer and Thompson, 1995). Hence pseudo-convergence is a better term.

1.11.3 One Long Run versus Many Short Runs

When you are in the black box situation, you have no idea how long runs need to be to get good mixing (convergence rather than pseudo-convergence). If you have a run that is already long enough, then an autocovariance plot like Figure 1.6 gives good information about mixing, and you know that you need to run a large multiple of the time it takes the

autocovariances to decay to nearly zero. But if all the runs you have done so far are nowhere near long enough, then they provide no information about how long is long enough.

The phenomenon of pseudo-convergence has led many people to the idea of comparing multiple runs of the sampler started at different points. If the multiple runs appear to converge to the same distribution, then—according to the multistart heuristic—all is well. But this assumes that you can arrange to have at least one starting point in each part of the state space to which the sampler can pseudo-converge. If you cannot do that—and in the black box situation you never can—then the multistart heuristic is worse than useless: it can give you confidence that all is well when in fact your results are completely erroneous.

Worse, addiction to many short runs can keep one from running the sampler long enough to detect pseudo-convergence or other problems, such as bugs in the code. People who have used MCMC in complicated problems can tell stories about samplers that appeared to be converging until, after weeks of running, they discovered a new part of the state space and the distribution changed radically. If those people had thought it necessary to make hundreds of runs, none of them could have been several weeks long.

Your humble author has a dictum that the least one can do is to make an overnight run. What better way for your computer to spend its time? In many problems that are not too complicated, this is millions or billions of iterations. If you do not make runs like that, you are simply not serious about MCMC. Your humble author has another dictum (only slightly facetious) that one should start a run when the paper is submitted and keep running until the referees' reports arrive. This cannot delay the paper, and may detect pseudo-convergence.

1.11.4 Burn-In

Burn-in is a colloquial term that describes the practice of throwing away some iterations at the beginning of an MCMC run. This notion says that you start somewhere, say at x, then you run the Markov chain for n steps (the burn-in period) during which you throw away all the data (no output). After the burn-in you run normally, using each iterate in your MCMC calculations.

The name "burn-in" comes from electronics. Many electronics components fail quickly. Those that do not are a more reliable subset. So a burn-in is done at the factory to eliminate the worst ones.

Markov chains do not work the same way. Markov chain "failure" (nonconvergence or pseudo-convergence) is different from electronic component failure. Running longer may cure the first, but a dead transistor is dead forever. Thus "burn-in" is a bad term in MCMC, but there is more wrong than just the word, there is something fishy about the whole concept.

Figure 1.7 illustrates the issue that burn-in addresses. It shows an AR(1) time series with all parameters except starting position the same as Figure 1.1 so the equilibrium distribution, normal with mean zero and variance (Equation 1.13), is the same for both. In Figure 1.7 the starting position is far out in the tail of the equilibrium distribution, 10 standard deviations from the mean. In Figure 1.1 the starting position is the mean (zero). It takes several hundred iterations before the sample path in Figure 1.7 gets into the region containing the whole sample path in Figure 1.1.

The naive idea behind burn-in is that if we throw away several hundred iterations from Figure 1.7 it will be just as good as Figure 1.1. Overgeneralizing examples like Figure 1.7 leads to the idea that every MCMC run should have burn-in. Examples like Figure 1.1 show that this is not so. A Markov chain started anywhere near the center of the equilibrium distribution needs no burn-in.

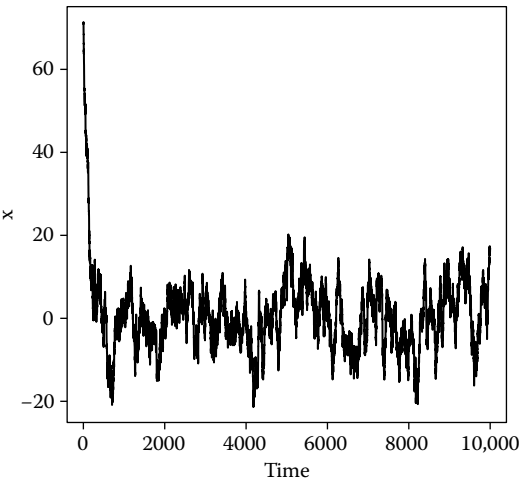

FIGURE 1.7
Time series plot for AR(1) example. Differs from Figure 1.1 only in the starting position.

> Burn-in is only one method, and not a particularly good method, of finding a good starting point.

There are several methods other than burn-in for finding a good starting point. One rule that is unarguable is

> Any point you don't mind having in a sample is a good starting point.

In a typical application, one has no theoretical analysis of the Markov chain dynamics that tells where the good starting points are (nor how much burn-in is required to get to a good starting point). All decisions about starting points are based on the output of some preliminary runs that appear to have "converged." Any point of the parts of these preliminary runs one believes to be representative of the equilibrium distribution is as good a starting point as any other.

So a good rule to follow is to start the next run where the last run ended. This is the rule most authorities recommend for random number generator seeds and the one used by R. It is also used by functions in the R package `mcmc` as discussed in Section 1.13 below.

Another method is to start at a mode of the equilibrium distribution (which can sometimes be found by optimization before doing MCMC) if it is known to be in a region of appreciable probability.

None of the examples in this chapter use burn-in. All use an alternative method of finding starting points. Burn-in is mostly harmless, which is perhaps why the practice persists. But everyone should understand that it is unnecessary, and those who do not use it are not thereby making an error.

Burn-in has a pernicious interaction with the multistart heuristic. If one believes in multistart, then one feels the need to start at many widely dispersed, and hence bad, starting points. Thus all of these short runs need be shortened some more by burn-in. Thus an erroneous belief in the virtues of multistart leads to an erroneous belief in the necessity of burn-in.

Another erroneous argument for burn-in is unbiasedness. If one could start with a realization from the equilibrium distribution, then the Markov chain would be stationary

and the Monte Carlo approximation (Equation 1.2) would be an unbiased estimator of what it estimates (Equation 1.1). Burn-in does not produce a realization from the equilibrium distribution, hence does not produce unbiasedness. At best it produces a small bias, but the alternative methods also do that. Moreover, the bias is of order n^{-1}, where n is the Monte Carlo sample size, whereas the MCSE is of order $n^{-1/2}$, so bias is negligible in sufficiently long runs.

1.11.5 Diagnostics

Many MCMC diagnostics have been proposed in the literature. Some work with one run of a Markov chain, but tell little that cannot be seen at a glance at a time series plot like Figure 1.1 or an autocorrelation plot like Figure 1.3. Others with multiple runs of a Markov chain started at different points, what we called the multistart heuristic above. Many of these come with theorems, but the theorems never prove the property you really want a diagnostic to have. These theorems say that if the chain converges, then the diagnostic will probably say that the chain converged, but they do not say that if the chain pseudo-converges, then the diagnostic will probably say that the chain did not converge. Theorems that claim to reliably diagnose pseudo-convergence have unverifiable conditions that make them useless. For example, as we said above, it is clear that a diagnostic based on the multistart heuristic will reliably diagnose pseudo-convergence if there is at least one starting point in each part of the state space to which the sampler can pseudo-converge, but in practical applications one has no way of arranging that.

There is only one perfect MCMC diagnostic: perfect sampling (Propp and Wilson, 1996; Kendall and Møller, 2000; see also Chapter 8, this volume). This is best understood as not a method of MCMC but rather a method of Markov-chain-assisted i.i.d. sampling. Since it is guaranteed to produce an i.i.d. sample from the equilibrium distribution of the Markov chain, a sufficiently large sample is guaranteed to not miss any parts of the state space having appreciable probability. Perfect sampling is not effective as a sampling scheme. If it works, then simply running the underlying Markov chain in MCMC mode will produce more accurate results in the same amount of computer time. Thus, paradoxically, perfect sampling is most useful when it fails to produce an i.i.d. sample of the requested size in the time one is willing to wait. This shows that the underlying Markov chain is useless for sampling, MCMC or perfect.

Perfect sampling does not work on black box MCMC (Section 1.11.1 above), because it requires complicated theoretical conditions on the Markov chain dynamics. No other diagnostic ever proposed works on black box MCMC, because if you know nothing about the Markov chain dynamics or equilibrium distribution except what you learn from output of the sampler, you can always be fooled by pseudo-convergence.

> There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we now know we don't know. But there are also unknown unknowns. These are things we do not know we don't know.
>
> **Donald Rumsfeld**
> *US Secretary of Defense*

Diagnostics can find the known unknowns. They cannot find the unknown unknowns. They cannot find out what a black box MCMC sampler will do eventually. Only sufficiently long runs can do that.

1.12 Elementary Theory of MCMC

We say that a bit of computer code that makes a pseudorandom change to its state is an *update mechanism*. We are interested in update mechanisms that preserve a specified distribution, that is, if the state has the specified distribution before the update, then it has the same distribution after the update. From them we can construct Markov chains to sample that distribution.

We say that an update mechanism is *elementary* if it is not made up of parts that are themselves update mechanisms preserving the specified distribution.

1.12.1 The Metropolis–Hastings Update

Suppose that the specified distribution (the desired stationary distribution of the MCMC sampler we are constructing) has *unnormalized density* h. This means that h is a positive constant times a probability density. Thus h is a nonnegative-valued function that integrates (for continuous state) or sums (for discrete state) to a value that is finite and nonzero. The *Metropolis–Hastings update* does the following:

- When the current state is x, propose a move to y, having conditional probability density given x denoted $q(x, \cdot)$.
- Calculate the *Hastings ratio*

$$r(x, y) = \frac{h(y)q(y, x)}{h(x)q(x, y)}. \tag{1.20}$$

- Accept the proposed move y with probability

$$a(x, y) = \min(1, r(x, y)), \tag{1.21}$$

  that is, the state after the update is y with probability $a(x, y)$, and the state after the update is x with probability $1 - a(x, y)$.

The last step is often called *Metropolis rejection*. The name is supposed to remind one of "rejection sampling" in OMC, but this is a misleading analogy because in OMC rejection sampling is done repeatedly until some proposal is accepted (so it always produces a new value of the state). In contrast, one Metropolis–Hastings update makes one proposal y, which is the new state with probability $a(x, y)$, but otherwise the new state the same as the old state x. Any attempt to make Metropolis rejection like OMC rejection, destroys the property that this update preserves the distribution with density h.

The Hastings ratio (Equation 1.20) is undefined if $h(x) = 0$, thus we must always arrange that $h(x) > 0$ in the initial state. There is no problem if $h(y) = 0$. All that happens is that $r(x, y) = 0$ and the proposal y is accepted with probability zero. Thus the Metropolis–Hastings update can never move to a new state x having $h(x) = 0$. Note that the proposal y must satisfy $q(x, y) > 0$ with probability one because $q(x, \cdot)$ is the conditional density of y given x. Hence, still assuming $h(x) > 0$, the denominator of the Hastings ratio is nonzero with probability one, and the Hastings ratio is well defined. Note that either term of the numerator of the Hastings ratio can be zero, so the proposal is almost surely rejected if

either $h(y) = 0$ or $q(y, x) = 0$, that is, if y is an impossible value of the desired equilibrium distribution or if x is an impossible proposal when y is the current state.

We stress that nothing bad happens if the proposal y is an impossible value of the desired equilibrium distribution. The Metropolis–Hastings update automatically does the right thing, almost surely rejecting such proposals. Hence, it is not necessary to arrange that proposals are always possible values of the desired equilibrium distribution; it is only necessary to assure that one's implementation of the unnormalized density function h works when given any possible proposal as an argument and gives $h(y) = 0$ when y is impossible.

If `unifrand` is a function with no arguments that produces one $U(0, 1)$ random variate and the Hastings ratio has already been calculated and stored in a variable `r`, then the following computer code does the Metropolis rejection step:

```
if (unifrand() < r) {
    x = y
}
```

The variable `x`, which is considered the state of the Markov chain, is set to `y` (the proposal) when a uniform random variate is less than the Hastings ratio `r` and left alone otherwise.

The following computer code works with the log Hastings ratio `logr` to avoid overflow:

```
if (logr >= 0 || unifrand() < exp(logr)) {
    x = y
}
```

It uses the "short circuit" property of the `||` operator in the R or C language. Its second operand `unifrand() < exp(logr)` is only evaluated when its first operand `logr >= 0` evaluates to `FALSE`. Thus `exp(logr)` can never overflow.

1.12.2 The Metropolis–Hastings Theorem

We now prove that the Metropolis–Hastings update is reversible with respect to h, meaning that the transition probability that describes the update is reversible with respect to the distribution having unnormalized density h.

If X_n is the current state and Y_n is the proposal, we have $X_n = X_{n+1}$ whenever the proposal is rejected. Clearly, the distribution of (X_n, X_{n+1}) given rejection is exchangeable.

Hence, it only remains to be shown that (X_n, Y_n) is exchangeable given acceptance. We need to show that

$$\mathrm{E}\{f(X_n, Y_n)a(X_n, Y_n)\} = \mathrm{E}\{f(Y_n, X_n)a(X_n, Y_n)\}$$

for any function f that has expectation (assuming X_n has desired stationary distribution). That is, we must show we can interchange arguments of f in

$$\iint f(x, y)h(x)a(x, y)q(x, y)\,dx\,dy \tag{1.22}$$

(with integrals replaced by sums if the state is discrete), and that follows if we can interchange x and y in

$$h(x)a(x, y)q(x, y) \tag{1.23}$$

because we can exchange x and y in Equation 1.22, x and y being dummy variables. Clearly only the set of x and y such that $h(x) > 0$, $q(x, y) > 0$, and $a(x, y) > 0$ contributes to the integral or (in the discrete case) sum (Equation 1.22), and these inequalities further imply that $h(y) > 0$ and $q(y, x) > 0$. Thus we may assume these inequalities, in which case we have

$$r(y, x) = \frac{1}{r(x, y)}$$

for all such x and y.

Suppose that $r(x, y) \le 1$, so $r(x, y) = a(x, y)$ and $a(y, x) = 1$. Then

$$\begin{aligned} h(x)a(x, y)q(x, y) &= h(x)r(x, y)q(x, y) \\ &= h(y)q(y, x) \\ &= h(y)q(y, x)a(y, x). \end{aligned}$$

Conversely, suppose that $r(x, y) > 1$, so $a(x, y) = 1$ and $a(y, x) = r(y, x)$. Then

$$\begin{aligned} h(x)a(x, y)q(x, y) &= h(x)q(x, y) \\ &= h(y)r(y, x)q(y, x) \\ &= h(y)a(y, x)q(y, x). \end{aligned}$$

In either case we can exchange x and y in Equation 1.23, and the proof is done.

1.12.3 The Metropolis Update

The special case of the Metropolis–Hastings update when $q(x, y) = q(y, x)$ for all x and y is called the *Metropolis update*. Then the Hastings ratio (Equation 1.20) simplifies to

$$r(x, y) = \frac{h(y)}{h(x)} \tag{1.24}$$

and is called the Metropolis ratio or the odds ratio. Thus Metropolis updates save a little time in calculating $r(x, y)$ but otherwise have no advantages over Metropolis–Hastings updates.

One obvious way to arrange the symmetry property is to make proposals of the form $y = x + e$, where e is stochastically independent of x and symmetrically distributed about zero. Then $q(x, y) = f(y - x)$, where f is the density of e. Widely used proposals of this type have e normally distributed with mean zero or e uniformly distributed on a ball or a hypercube centered at zero (see Section 1.12.10 below for more on such updates).

1.12.4 The Gibbs Update

In a *Gibbs update* the proposal is from a conditional distribution of the desired equilibrium distribution. It is always accepted.

The proof of the theorem that this update is reversible with respect to the desired equilibrium distribution is trivial. Suppose that X_n has the desired stationary distribution. Suppose that the conditional distribution of X_{n+1} given $f(X_n)$ is same as the conditional distribution of X_n given $f(X_n)$. Then the pair (X_n, X_{n+1}) is conditionally exchangeable given $f(X_n)$, hence unconditionally exchangeable.

In common parlance, a Gibbs update uses the conditional distribution of one component of the state vector given the rest of the components, that is, the special case of the update described above where $f(X_n)$ is X_n with one component omitted. Conditional distributions of this form are called "full conditionals." There is no reason other than tradition why such conditional distributions should be preferred.

In fact other conditionals have been considered in the literature. If $f(X_n)$ is X_n with several components omitted, this is called "block Gibbs." Again, there is no reason other than tradition why such conditional distributions should be preferred.

If one insists that Gibbs update only apply to full conditionals, then one could call the updates described here "generalized Gibbs." But the "generalized" here is not much of a generalization. Simply do a change of variable so that $f(X_n)$ is a group of components of the new state vector and "generalized Gibbs" is "block Gibbs." Also the argument for all these updates is exactly the same.

Gibbs updates have one curious property not shared by other Metropolis–Hastings updates: they are *idempotent*, meaning the effect of multiple updates is the same as the effect of just one. This is because the update never changes $f(X_n)$, hence the result of many repetitions of the same Gibbs update results in X_{n+1} having the conditional distribution given $f(X_n)$ just like the result of a single update. In order for Gibbs elementary updates to be useful, they must be combined somehow with other updates.

1.12.5 Variable-at-a-Time Metropolis–Hastings

Gibbs updates alter only part of the state vector; when using "full conditionals" the part is a single component. Metropolis–Hastings updates can be modified to do the same.

Divide the state vector into two parts, $x = (u, v)$. Let the proposal alter u but not v. Hence, the proposal density has the form $q(x, u)$ instead of the $q(x, y)$ we had in Section 1.12.1. Again let $h(x) = h(u, v)$ be the unnormalized density of the desired equilibrium distribution. The variable-at-a-time Metropolis–Hastings update does the following:

- When the current state is $x = (u, v)$, propose a move to $y = (u^*, v)$, where u^* has conditional probability density given x denoted $q(x, \cdot) = q(u, v, \cdot)$.
- Calculate the *Hastings ratio*

$$r(x, y) = \frac{h(u^*, v)q(u^*, v, u)}{h(u, v)q(u, v, u^*)}.$$

- Accept the proposed move y with probability (Equation 1.21), that is, the state after the update is y with probability $a(x, y)$, and the state after the update is x with probability $1 - a(x, y)$.

We shall not give a proof of the validity of variable-at-a-time Metropolis–Hastings, which would look very similar to the proof in Section 1.12.2.

The term "variable-at-a-time Metropolis–Hastings" is something of a misnomer. The sampler run in Metropolis et al. (1953) was a "variable-at-a-time" sampler. For historical accuracy, the name "Metropolis algorithm" should include the updates described in Section 1.12.1 and in this section. Current usage, however, seems otherwise, naming the samplers as we have done here.

1.12.6 Gibbs Is a Special Case of Metropolis–Hastings

To see that Gibbs is a special case of Metropolis–Hastings, do a change of variable so that the new state vector can be split $x = (u, v)$ as we did in the preceding section, and v is the part of the state on which the Gibbs update conditions. Thus we are doing block Gibbs updating u from its conditional distribution given v. Factor the unnormalized density $h(u, v) = g(v)q(v, u)$, where $g(v)$ is an unnormalized marginal of v and $q(v, u)$ is the (properly normalized) conditional of u given v. Now do a Metropolis–Hastings update with q as the proposal distribution. The proposal is $y = (u^*, v)$, where u^* has the distribution $q(v, \cdot)$. The Hastings ratio is

$$r(x, y) = \frac{h(u^*, v)q(u, v)}{h(u, v)q(v, u^*)} = \frac{g(v)q(v, u^*)q(u, v)}{g(v)q(v, u)q(v, u^*)} = 1.$$

Hence the proposal is always accepted.

1.12.7 Combining Updates

1.12.7.1 Composition

Let $P_1, \ldots, P_k$ be update mechanisms (computer code) and let $P_1P_2 \cdots P_k$ denote the composite update that consists of these updates done in that order with P_1 first and P_k last. If each P_i preserves a distribution, then obviously so does $P_1P_2 \ldots P_k$.

If $P_1, \ldots, P_k$ are the Gibbs updates for the "full conditionals" of the desired equilibrium distribution, then the composition update is often called a *fixed scan Gibbs sampler*.

As a simple example, suppose that the desired equilibrium distribution is exchangeable and multivariate normal. Then the conditional distribution of one component of the state vector given the rest is univariate normal with mean that is a symmetric linear function of the rest of the components and constant variance. In the special case where there are just two components, the fixed scan Gibbs sampler is just consecutive pairs of an AR(1) process (Section 1.9 above).

1.12.7.2 Palindromic Composition

Note that $P_1P_2 \ldots P_k$ is not reversible with respect to the distribution it preserves unless the transition probabilities associated with $P_1P_2 \ldots P_k$ and $P_kP_{k-1} \ldots P_1$ are the same.

The most obvious way to arrange reversibility is to make $P_i = P_{k-i}$, for $i = 1, \ldots, k$. Then we call this composite update *palindromic*. Palindromic compositions are reversible, nonpalindromic ones need not be.

1.12.8 State-Independent Mixing

Let P_y be update mechanisms (computer code) and let $\mathrm{E}(P_Y)$ denote the update that consists of doing a random one of these updates: generate Y from some distribution and do P_Y.

If Y is independent of the current state and each P_y preserves the same distribution, then so does $\mathrm{E}(P_Y)$. If X_n has the desired equilibrium distribution, then it also has this distribution conditional on Y, and X_{n+1} also has this distribution conditional on Y. Since the conditional distribution of X_{n+1} does not depend on Y, these variables are independent, and X_{n+1} has the desired equilibrium distribution unconditionally.

Furthermore, the Markov chain with update $\mathrm{E}(P_Y)$ is reversible if each P_y is reversible.

"Mixture" is used here in the sense of mixture models. The update $\mathrm{E}(P_Y)$ is the mixture of updates P_y.

The most widely used mixtures use a finite set of y values. For example, one popular way to combine the "full conditional" Gibbs updates, one for each component of the state vector, is by state-independent mixing using the uniform distribution on the set of full conditionals as the mixing distribution. This is often called a *random scan Gibbs sampler*. The choice of the uniform distribution is arbitrary. It has no optimality properties. It does, however, make a simple default choice.

Mixing and composition can be combined. Suppose we have elementary update mechanisms $P_1, \ldots, P_k$, and let $\mathcal{Y}$ be a set of functions from $\{1, \ldots, m\}$ to $\{1, \ldots, k\}$. For $y \in \mathcal{Y}$, let Q_y denote the composition $P_{y(1)} P_{y(2)} \ldots P_{y(m)}$. Now consider the update $\mathrm{E}(Q_Y)$, where Y is a random element of $\mathcal{Y}$ independent of the state of the Markov chain.

If $m = k$ and the P_i are the "full conditional" Gibbs updates and Y has the uniform distribution on $\mathcal{Y}$, which consists of all permutations of $1, \ldots, k$, then this mixture of compositions sampler is often called a *random sequence scan Gibbs sampler*.

We are not fond of this "scan" terminology, because it is too limiting. It focuses attention on a very few special cases of combination by composition and mixing, special cases that have no optimality properties and no reason other than tradition for their prominence.

State-independent mixing with the mixing distribution having an infinite sample space has also been used. Bélisle et al. (1993) and Chen and Schmeiser (1993) investigate the "hit and run algorithm" which uses elementary updates P_y where the state space of the Markov chain is Euclidean and y is a direction in the state space. Do a change of coordinates so that y is a coordinate direction, and do a Gibbs or other variable-at-a-time Metropolis–Hastings update of the coordinate in the y direction. The mixture update $\mathrm{E}(P_Y)$ is called a "hit and run sampler" when Y has the uniform distribution on directions.

Again there is no particular reason to use a "hit and run" sampler. It is merely one of an infinite variety of samplers using composition and state-independent mixing.

State-dependent mixing is possible, but the argument is very different (Section 1.17.1 below).

1.12.9 Subsampling

Another topic that is not usually discussed in terms of composition and mixing, although it is another special case of them, is subsampling of Markov chains.

If P is an update mechanism, we write P^k to denote the k-fold composition of P with itself. If $X_1, X_2, \ldots$ is a Markov chain with update mechanism P, then $X_1, X_{k+1}, X_{2k+1}, \ldots$ is a Markov chain with update mechanism P^k.

The process that takes every kth element of a Markov chain $X_1, X_2, \ldots$ forming a new Markov chain $X_1, X_{k+1}, X_{2k+1}, \ldots$ is called *subsampling* the original Markov chain at *spacing* k. As we just said, the result is another Markov chain. Hence, a subsampled Markov chain is just like any other Markov chain.

According to Elizabeth Thompson, "You don't get a better answer by throwing away data." This was proved as a theorem about Markov chains by Geyer (1992) for reversible Markov chains and by MacEachern and Berliner (1994) for nonreversible Markov chains. Subsampling cannot improve the accuracy of MCMC approximation; it must make things worse.

The original motivation for subsampling appears to have been to reduce autocorrelation in the subsampled chain to a negligible level. Before 1994 the Markov chain CLT was not well understood by statisticians, so appeal was made to a non-theorem: the central limit

almost-but-not-quite theorem for almost-but-not-quite i.i.d. data. Now that the Markov chain CLT is well understood, this cannot be a justification for subsampling.

Subsampling may appear to be necessary just to reduce the amount of output of a Markov chain sampler to manageable levels. Billions of iterations may be needed for convergence, but billions of iterations of output may be too much to handle, especially when using R, which chokes on very large objects. But nonoverlapping batch means (Section 1.10.1) can reduce the size of the output with no loss of accuracy of estimation. Moreover, one does not need to know the batch length necessary to make the empirical variance of the batch means a good estimate of the asymptotic variance in the Markov chain CLT in order to use batches to reduce the size of output. The method of Section 1.10.3 allows one to use batches that are too short and still obtain accurate estimates of the asymptotic variance in the Markov chain CLT. Hence, if the objective is to reduce the size of output, batching is better than subsampling.

Hence, the only reason to use subsampling is to reduce the size of output when one cannot use batching. Good MCMC code, for example the functions `metrop` and `temper` in the R contributed package `mcmc` (Geyer, 2010b), allow an arbitrary function g supplied by the user as an R function to be used in calculation of the batch means in Equation 1.16. Other MCMC code that does not allow this may not output batch means for required functionals of the Markov chain. In this case the only way to reduce the size of output and still calculate the required functionals is subsampling. Another case where one cannot use the batch means is when the required functionals are not known when the sampling is done. This occurs, for example, in Monte Carlo likelihood approximation (Geyer and Thompson, 1992).

Geyer (1992) gave another justification of subsampling based on the cost of calculating the function g in a functional (Section 1.6 above). If the cost in computing time of calculating $g(X_i)$ is much more than the cost of sampling (producing X_i given X_{i-1}), then subsampling may be justified. This is rarely the case, but it does happen.

1.12.10 Gibbs and Metropolis Revisited

Our terminology of "elementary updates" combined by "composition" or "mixing" or both is not widespread. The usual terminology for a much more limited class of samplers is the following:

- A *Gibbs sampler* is an MCMC sampler in which all of the elementary updates are Gibbs, combined either by composition (fixed scan), by mixing (random scan), or both (random sequence scan), the "scan" terminology being explained in Section 1.12.8 above.
- A *Metropolis algorithm* is an MCMC sampler in which all of the elementary updates are Metropolis, combined either by composition, mixing, or both (and the same "scan" terminology is used).
- A *Metropolis–Hastings algorithm* is an MCMC sampler in which all of the elementary updates are Metropolis–Hastings, combined either by composition, mixing, or both (and the same "scan" terminology is used).
- A *Metropolis-within-Gibbs sampler* is the same as the preceding item. This name makes no sense at all since Gibbs is a special case of Metropolis–Hastings (Section 1.12.6 above), but it is widely used.

- An *independence Metropolis–Hastings algorithm* (named by Tierney, 1994) is a special case of the Metropolis–Hastings algorithm in which the proposal distribution does not depend on the current state: $q(x, \cdot)$ does not depend on x.
- A *random-walk Metropolis–Hastings algorithm* (named by Tierney, 1994) is a special case of the Metropolis–Hastings algorithm in which the proposal has the form $x + e$, where e is stochastically independent of the current state x, so $q(x, y)$ has the form $f(y - x)$.

The Gibbs sampler became very popular after the paper of Gelfand and Smith (1990) appeared. The term MCMC had not been coined (Geyer, 1992). It was not long, however, before the limitations of the Gibbs sampler were recognized. Peter Clifford (1993), discussing Smith and Roberts (1993), Besag and Green (1993), and Gilks et al. (1993), said:

> Currently, there are many statisticians trying to reverse out of this historical *cul-de-sac*. To use the Gibbs sampler, we have to be good at manipulating conditional distributions . . . this rather brings back the mystique of the statisticians.

The American translation of "reverse out of this *cul-de-sac*" is "back out of this blind alley." Despite this, many naive users still have a preference for Gibbs updates that is entirely unwarranted. If I had a nickel for every time someone had asked for help with slowly converging MCMC and the answer had been to stop using Gibbs, I would be rich. Use Gibbs updates only if the resulting sampler works well. If not, use something else.

One reason sometimes given for the use of Gibbs updates is that they are "automatic." If one chooses to use a Gibbs sampler, no other choices need be made, whereas if one uses the Metropolis–Hastings algorithm, one must choose the proposal distribution, and even if one's choice of Metropolis–Hastings algorithm is more restricted, say to normal random-walk Metropolis–Hastings, there is still the choice of the variance matrix of the normal proposal distribution. This "automaticity" of the Gibbs sampler is illusory, because even if one only knows about "scans" one still must choose between fixed and random scan. Moreover, one should consider "block Gibbs" or even the more general Gibbs updates described in Section 1.12.4 above.

Nevertheless, Gibbs does seem more automatic than Metropolis–Hastings to many users. The question is whether this lack of options is a good thing or a bad thing. It is good if it works well and bad otherwise.

1.13 A Metropolis Example

We now turn to a realistic example of MCMC, taken from the package vignette of the `mcmc` contributed R package (Geyer, 2010b). The function `metrop` in this package runs a normal random-walk Metropolis sampler in the terminology of Section 1.12.10 having equilibrium distribution for a continuous random vector specified by a user-written R function that calculates its log unnormalized density. A major design goal of this package is that there be very little opportunity for user mistakes to make the simulation incorrect. For the `metrop` function, if the user codes the log unnormalized density function correctly, then the function will run a Markov chain having the correct stationary distribution (specified by this user-written function). There is nothing other than incorrectly writing the log unnormalized

density function that the user can do to make the Markov chain have the wrong stationary distribution.

It may seem that this is a very weak correctness property. There is no guarantee that the Markov chain mixes rapidly and so produces useful results in a reasonable amount of time. But nothing currently known can guarantee that for arbitrary problems. Methods of proving rapid mixing, although they are applicable in principle to arbitrary problems, are so difficult that they have actually been applied only to a few simple examples. Moreover, they are entirely pencil-and-paper proofs. There is nothing the computer can do to assure rapid mixing of Markov chains for arbitrary user-specified equilibrium distributions. Thus this weak correctness property (having the correct equilibrium distribution) is the most one can expect a computer program to assure.

Thus this "weak" correctness property is the strongest property one can reasonably assert for an MCMC program. All MCMC programs should guarantee it, but how many do? The functions in the `mcmc` package have been exhaustively tested using the methodology explained in Section 1.16 below and further described in the package vignette `debug.pdf` that comes with every installation of the package. All of the tests are in the `tests` directory of the source code of the package, which is available from CRAN (http://www.cran.r-project.org/).

In addition to an R function that specifies the log unnormalized density of the equilibrium distribution, the user may also provide an R function that specifies an arbitrary functional of the Markov chain to be output. If the Markov chain is $X_1, X_2, \ldots$ and this user-supplied R function codes the mathematical function g, then $g(X_1), g(X_2), \ldots$ is output. Alternatively, batch means of $g(X_1), g(X_2), \ldots$ are output.

Finally, the user must specify the variance matrix of the multivariate normal distribution used in the "random-walk" proposal. There is nothing else the user can do to affect the Markov chain simulated by the `metrop` function.

Let us see how it works. We use the example from the package vignette `demo.pdf` that comes with every installation of the package. This is a Bayesian logistic regression problem that uses the data set `logit` in the package. There are five variables in this data frame, the response `y` and four quantitative predictor variables `x1`, `x2`, `x3`, and `x4`.

A frequentist analysis of these data is done by the following R statements:

```
library(mcmc)
data(logit)
out <- glm(y ~ x1 + x2 + x3 + x4, data = logit,
    family = binomial(), x = TRUE)
summary(out)
```

We wish to do a Bayesian analysis where the prior distribution for the five regression coefficients (one for each predictor and an intercept) makes them i.i.d. normal with mean 0 and standard deviation 2.

The log unnormalized posterior (log likelihood plus log prior) density for this model is calculated by the R function `lupost` defined as follows:

```
x <- out$x
y <- out$y

lupost <- function(beta, x, y, ...) {
    eta <- as.numeric(x %*% beta)
```

```
    logp <- ifelse(eta < 0, eta - log1p(exp(eta)),- log1p
     (exp(- eta)))
    logq <- ifelse(eta < 0, - log1p(exp(eta)), - eta - log1p
     (exp(- eta)))
    logl <- sum(logp[y == 1]) + sum(logq[y == 0])
    return(logl - sum(beta^2) / 8)
}
```

This assumes that `out` is the result of the call to `glm` shown above, so `y` is the response vector and `x` is the model matrix for this logistic regression.

The tricky calculation of the log likelihood avoids overflow and catastrophic cancelation in calculation of $\log(p)$ and $\log(q)$, where

$$p = \frac{\exp(\eta)}{1 + \exp(\eta)} = \frac{1}{1 + \exp(-\eta)},$$
$$q = \frac{1}{1 + \exp(\eta)} = \frac{\exp(-\eta)}{1 + \exp(-\eta)},$$

so taking logs gives

$$\log(p) = \eta - \log(1 + \exp(\eta)) = -\log(1 + \exp(-\eta)),$$
$$\log(q) = -\log(1 + \exp(\eta)) = -\eta - \log(1 + \exp(-\eta)).$$

To avoid overflow, we always chose the case where the argument of exp is negative. We have also avoided catastrophic cancelation when $|\eta|$ is large. If η is large and positive, then

$$p \approx 1,$$
$$q \approx 0,$$
$$\log(p) \approx -\exp(-\eta),$$
$$\log(q) \approx -\eta - \exp(-\eta),$$

and our use of the R function `log1p`, which calculates the function $x \mapsto \log(1 + x)$ correctly for small x, avoids problems with calculating $\log(1 + \exp(-\eta))$ here. The case where η is large and negative is similar. The above definitions having been made, the following statements do an MCMC run:

```
beta.init <- as.numeric(coefficients(out))
out <- metrop(lupost, beta.init, 1e3, x = x, y = y)
```

where `beta.init` is the initial state of the Markov chain (it would be more natural to a Bayesian to use the posterior mode rather than the maximum likelihood estimate, but the starting position makes no difference so long as it is not too far out in the tails of the equilibrium distribution) and where `1e3` is the MCMC sample size. The default batch length is one, so there is no batching here. The component `out$accept` of the result gives the acceptance rate (the fraction of Metropolis updates in which the proposal is accepted) and the component `out$batch` gives the output of the Markov chain as an $n \times p$ matrix, where n is the number of iterations here where there is no batching (although in general

n is the number of batches) and where p is the dimension of the state space here where no functional of the Markov chain is specified and the default is the identity functional (although in general p is the dimension of the result of the user-supplied output function).

The functions in the `mcmc` package are designed so that if given the output of a preceding run as their first argument, they continue the run of the Markov chain where the other run left off. For example, if we were to say

```
out2 <- metrop(out, x = x, y = y)
```

here, then `rbind(out$batch, out2$batch)` would be a run of the Markov chain. The second invocation of the `metrop` function starts with the seed of R's random number generator (RNG) and the state of the Markov chain set to what they were when the first invocation finished. Thus there is no difference between `rbind(out$batch, out2$batch)` and the result of one invocation starting at the same RNG seed and initial state and running for twice as many iterations as the two shown here did.

This "restart" property obviates any need for burn-in. If the first run "converged" in the sense that any part of the run was in a high-probability part of the state space, then the second run starts in a good place and needs no burn-in. Since the first run started at the maximum likelihood estimate, which is in a high-probability part of the state space, the first run needed no burn-in either.

Using this function is not quite this simple. We need to adjust the normal proposal to achieve a reasonable acceptance rate. It is generally accepted (Gelman et al., 1996) that an acceptance rate of about 20% is right, although this recommendation is based on the asymptotic analysis of a toy problem (simulating a multivariate normal distribution) for which one would never use MCMC and is very unrepresentative of difficult MCMC applications. Geyer and Thompson (1995) came to a similar conclusion, that a 20% acceptance rate is about right, in a very different situation. But they also warned that a 20% acceptance rate could be very wrong, and produced an example where a 20% acceptance rate was impossible and attempting to reduce the acceptance rate below 70% would keep the sampler from ever visiting part of the state space. So the 20% magic number must be considered like other rules of thumb we teach in introductory courses (such as $n > 30$ means the normal approximation is valid). We know these rules of thumb can fail. There are examples in the literature where they do fail. We keep repeating them because we want something simple to tell beginners, and they are all right for some problems.

The `scale` argument to the `metrop` function specifies the variance matrix for the proposal. The default is the identity matrix. This results in too low an acceptance rate in this problem (0.008). A little bit of trial and error (shown in the vignette) shows that

```
out <- metrop(out, scale = 0.4, x = x, y = y)
```

gives about 20% acceptance rate, so this scaling, which specifies proposal variance matrix 0.4 times the identity matrix, is what we use. More complicated specification of the proposal variance is possible; see the help for the `metrop` function for details.

Now we do a longer run

```
out <- metrop(out, nbatch = 1e4, x = x, y = y)
```

and look at time series plots and autocorrelation plots (shown in the vignette), which show that the Markov chain seems to mix well and that autocorrelations are negligible after

lag 25. We use batch length 100 to be safe. We are interested here in calculating both posterior means and posterior variances. Variances are not functionals of the Markov chain, but squares are, and we can use the identity $\mathrm{var}(Z) = \mathrm{E}(Z^2) - \mathrm{E}(Z)^2$ to calculate variances from means and means of squares. Thus we run the following:

```
out <- metrop(out, nbatch = 1e2, blen = 100,
    outfun = function(z, ...) c(z, z^2), x = x, y = y)
```

Here the user-specified output function (argument `outfun` of the `metrop` function) maps the state `z`, a vector of length 5, to `c(z, z^2)`, a vector of length 10. So now `out$batch` is a 100×10 matrix, 100 being the number of batches (argument `nbatch`) and 10 being the length of the result of `outfun`).

Now

```
foo <- apply(out$batch, 2, mean)
foo.mcse <- apply(out$batch, 2, sd) / sqrt(out$nbatch)
```

are estimates of the posterior means of the components of the vector returned by `outfun` (the regression coefficients and their squares) and the MCSE of these estimates, respectively. The first five components are useful directly:

```
mu <- foo[1:5]
mu.mcse <- foo.mcse[1:5]
```

These are estimates of the posterior means of the regression coefficients and their MCSE (see the vignette for actual numbers).

Monte Carlo estimates of the posterior variances are found using $\mathrm{var}(Z) = \mathrm{E}(Z^2) - \mathrm{E}(Z)^2$,

```
sigmasq <- foo[6:10] - foo[1:5]^2
```

but to calculate the MCSE we need the delta method. Let u_i denote the sequence of batch means for one parameter and $\bar{u}$ the grand mean of this sequence (the estimate of the posterior mean of that parameter), let v_i denote the sequence of batch means for the squares of the same parameter and $\bar{v}$ the grand mean of that sequence (the estimate of the posterior second absolute moment of that parameter), and let $\mu = \mathrm{E}(\bar{u})$ and $\nu = \mathrm{E}(\bar{v})$. Then the delta method linearizes the nonlinear function

$$g(\mu, \nu) = \nu - \mu^2$$

as

$$\Delta g(\mu, \nu) = \Delta\nu - 2\mu\Delta\mu,$$

saying that

$$g(\bar{u}, \bar{v}) - g(\mu, \nu)$$

has the same asymptotic normal distribution as

$$(\bar{v} - \nu) - 2\mu(\bar{u} - \mu)$$

which, of course, has variance `1 / out$nbatch` times that of

$$(v_i - \nu) - 2\mu(u_i - \mu),$$

and this variance is estimated by

$$\frac{1}{n_{\text{batch}}} \sum_{i=1}^{n_{\text{batch}}} \bigl[(v_i - \bar{v}) - 2\bar{u}(u_i - \bar{u})\bigr]^2.$$

So

```
u <- out$batch[ , 1:5]
v <- out$batch[ , 6:10]
ubar <- apply(u, 2, mean)
vbar <- apply(v, 2, mean)
deltau <- sweep(u, 2, ubar)
deltav <- sweep(v, 2, vbar)
foo <- sweep(deltau, 2, ubar, "*")
sigmasq.mcse <- sqrt(apply((deltav - 2 * foo)^2,
2, mean) / out$nbatch)
```

does the MCSE for the posterior variance (see the vignette for actual numbers).

Another application of the delta method gives MCSE for posterior standard deviations (see the vignette for details).

1.14 Checkpointing

The "restart" property of the `metrop` and `temper` functions is also useful for checkpointing. If one wants to do very long runs, they need not be done with one function invocation. Suppose that `out` is the result of an invocation of `metrop` and that the log unnormalized density function and output function (if present) do not take additional arguments, getting any additional data from the R global environment, and suppose that any such additional data has been set up. Let `ncheck` be the number of repetitions of `out` we want to make. Then

```
for (icheck in 1:ncheck) {
    out <- metrop(out)
    save(out, file = sprintf("check%03d.rda", icheck))
}
```

does them and saves them on disk, unless the computer crashes for some reason. After a crash, only the work not done and saved is left to do. Set up any required global variables and `ncheck` as before, and restart with

```
files <- system("ls check*.rda", intern = TRUE)
kcheck <- length(files)
```

```
load(file = files[kcheck])
if (kcheck < ncheck) {
    for (icheck in (kcheck + 1):ncheck) {
        out <- metrop(out)
        save(out, file = sprintf("check%03d.rda", icheck))
    }
}
```

(this is for UNIX, e.g., Linux or MAC OS X, and would have to be modified for Microsoft Windows). When finished collect the results with

```
files <- system("ls check*.rda", intern = TRUE)
ncheck <- length(files)
batch <- NULL
for (icheck in 1:ncheck) {
    load(file = files[icheck])
    batch <- rbind(batch, out$batch, deparse.level = 0)
}
```

and batch is the same as `out$batch` from one long run. This idiom allows very long runs even with unreliable computers.

1.15 Designing MCMC Code

Nothing is easier than designing MCMC algorithms. Hundreds have been introduced into the literature under various names. All that are useful in non-toy problems are special cases of the Metropolis–Hastings–Green algorithm.

When one invents a new sampler, how does one argue that it is correct? One proves a theorem: the new sampler is a special case of the MHG algorithm. The proof is usually not difficult but does require tight reasoning, like all proofs. One common error is sloppiness about what is the state of the Markov chain. Many have made the mistake of having proposals depend on some variables in the computer program that are not considered part of the state in calculating the Hastings ratio, that is, the state space is considered one thing in one part of the argument and another thing in another part—a clear error if one thinks about it.

One does not have to call this theorem a theorem, but one does need the care in proving it that any theorem requires. A few hours of careful thought about what is and what is not a special case of the MHG algorithm can save weeks or months of wasted work on a mistake. This notion that you have to prove a theorem every time you invent an MCMC algorithm came to your humble author from the experience of humbling mistakes committed by himself and others. If you think you have to prove a theorem, you will (hopefully) exercise appropriately careful argument. If you think you can use your intuition, many sad stories could be told about failure of intuition. The MHG algorithm is not difficult but is also not very intuitive.

Before one can prove a theorem, one must state the theorem, and here too care is required. The theorem must state precisely how one's MCMC algorithm works, with no vagueness.

This is very important. One cannot correctly implement an MCMC algorithm in computer code when one has to guess what the algorithm actually is. Most erroneous MCMC algorithms (just like most erroneous attempts at theorems) result from vagueness.

These general remarks having been made, we now turn to some desirable features of MCMC code that few computer packages have but the `mcmc` package has shown to be very useful.

The first is the "restart" property discussed in Sections 1.13 and 1.14 above and possessed by both the `metrop` and `temper` functions. This is the property that the R object output by a function doing MCMC (or the equivalent object for computer languages other than R) should contain the RNG seeds and the final state of the Markov chain, so the next run can simply continue this run. A sampler with the "restart" property needs no burn-in (Section 1.11.4 above) and is easily checkpointed (Section 1.14).

The second is the property of outputting batch means for batches of a possibly subsampled chain, also possessed by both the `metrop` and `temper` functions, specified by the arguments `blen` and `nspac`. This property allows very long runs without overly voluminous output. If `nspac = 1` (the default, meaning no subsampling) is used, then no information is lost by the batching. The batches can be used for valid inference—regardless of whether the batch length is long enough for the ordinary method of batch means to work—as described in Section 1.10.3 above.

The third is the property of outputting batch means (for batches of a possibly subsampled chain) for an arbitrary functional of the Markov chain. The `mcmc` and `temper` functions do this via a user-specified function supplied as their `outfun` argument. This allows users to make the inferences they want without rewriting the R package. This makes statistical computer languages in which functions are not first-class objects (like they are in R) unsuitable for MCMC.

1.16 Validating and Debugging MCMC Code

Along with "black box" MCMC (Section 1.11.1) above we introduce the notion of "black box" testing of MCMC code. Black box testing is widely used terminology in software testing. It refers to tests that do not look inside the code, using only its ordinary input and output. Not looking at the code means it cannot use knowledge of the structure of the program or the values any of its internal variables. For MCMC code black box testing means you run the sampler and test that the output has the expected probability distribution.

Since goodness-of-fit testing for complicated multivariate probability distributions is very difficult, black box testing of MCMC code is highly problematic. It is even more so when the sampler is itself black box, so nothing is known about the expected equilibrium distribution except what we may learn from the sampler itself. Thus your humble author has been driven to the conclusion that black box testing of MCMC code is pointless.

Instead testing of the functions `metrop` and `temper` in the `mcmc` package uses a "white box" approach that exposes all important internal variables of the program when the optional argument `debug = TRUE` is specified. In particular, all uniform or normal random variates obtained from R's RNG system are output. This means that, assuming we can trust R's normal and uniform RNG, we can test whether `metrop` and `temper` behave properly as deterministic functions of those pseudorandom numbers obtained from R.

Testing whether a program correctly implements a deterministic function is much easier than testing whether it correctly simulates a specified probability distribution. In addition, when `debug = TRUE` these programs also output proposals, log Hastings ratios, and decisions in the Metropolis rejection step, making it easy to check whether these are correct and hence whether the Metropolis–Hastings algorithm is implemented correctly.

It must be admitted that, although this "white box" testing methodology it much superior to anything your humble author has previously used, it is not guaranteed to find conceptual problems. That is why a clearly written specification (what we called the "theorem" in the preceding section) is so important. During the writing of this chapter just such a conceptual bug was discovered in the `temper` function in versions of the `mcmc` package before 0.8. The terms $q(i, j)$ and $q(j, i)$ in the Hastings ratio for serial tempering (Equation 11.11 in Chapter 11, this volume) were omitted from the code, and the tests of whether the Hastings ratio was calculated correctly were implemented by looking at the code rather than the design document (the file `temper.pdf` in the `doc` directory of every installation of the `mcmc` package), which was correct.

Ideally, the tests should be implemented by someone other than the programmer of the code, a well-recognized principle in software testing. We know of no statistics code that conforms to this practice, perhaps because there is no tradition of refereeing computer code as opposed to papers. The most we can claim is that the "white box" testing methodology used for the `mcmc` would at least make such referring possible.

1.17 The Metropolis–Hastings–Green Algorithm

There are so many ideas in Green (1995) it is hard to know where to start. They include the following:

- State-dependent mixing of updates
- Measure-theoretic Metropolis–Hastings using Radon–Nikodym derivatives
- Per-update augmentation of the state space
- Metropolis–Hastings with Jacobians

any one of which would have been a major contribution by itself.

We have deferred discussion of the MHG algorithm till now because we wanted to avoid measure theory as long as we could. The MHG algorithm cannot easily be discussed without using measure-theoretic terminology and notation.

A kernel $K(x, A)$ is a generalization of regular conditional probability. For a fixed point x in the state space, $K(x, \cdot)$ is a countably-additive real signed measure on the state space. For a fixed measurable set A in the state space, $K(\cdot, A)$ is a measurable real-valued function on the state space. If

$$K(x, A) \geq 0, \quad \text{for all } x \text{ and } A,$$

then we say that K is *nonnegative*. If K is nonnegative and

$$K(x, A) \leq 1, \quad \text{for all } x \text{ and } A,$$

then we say that K is *sub-Markov*. If K is sub-Markov and

$$K(x, S) = 1, \quad \text{for all } x,$$

where S is the state space, then we say that K is *Markov*. A Markov kernel is a regular conditional probability and can be used to describe an elementary update mechanism for a Markov chain or a combined update. In widely used sloppy notation, we write

$$K(x, A) = \Pr(X_{t+1} \in A \mid X_t = x)$$

to describe the combined update (the sloppiness is the conditioning on an event of measure zero).

A kernel K is *reversible* with respect to a signed measure m if

$$\iint g(x)h(y)m(dx)K(x, dy) = \iint h(x)g(y)m(dx)K(x, dy)$$

for all measurable functions g and h such that the expectations exist. A Markov kernel P *preserves* a probability measure π if

$$\iint g(y)\pi(dx)P(x, dy) = \int g(x)\pi(dx)$$

for every bounded function g. Reversibility with respect to π implies preservation of π.

1.17.1 State-Dependent Mixing

Suppose we have a family of updates represented by Markov kernels P_i, $i \in I$. Choose one at random with probability $c_i(x)$ that depends on the current state x, and use it to update the state. The kernel that describes this combined update is

$$P(x, A) = \sum_{i \in I} c_i(x)P_i(x, A).$$

It is *not a theorem* that if each P_i preserves π, then P preserves π. The argument in Section 1.12.8 above does not work.

Define

$$K_i(x, A) = c_i(x)P_i(x, A).$$

If each K_i is reversible with respect to π, then the mixture kernel

$$P(x, A) = \sum_{i \in I} c_i(x)P_i(x, A) = \sum_{i \in I} K_i(x, A)$$

is reversible with respect to π and hence preserves π. This is how state-dependent mixing works.

It is often convenient to allow the identity kernel defined by

$$I(x, A) = \begin{cases} 1, & x \in A, \\ 0, & x \notin A, \end{cases}$$

to be among the P_i. The identity kernel is a Markov kernel that describes a do-nothing update (the state is the same before and after).

Sometimes state-dependent mixing involving the identity kernel is described differently. We insist that

$$c_i(x) \geq 0, \quad \text{for all } i \text{ and } x,$$

and

$$\sum_{i \in I} c_i(x) \leq 1, \quad \text{for all } x.$$

Then when x is the current state the mixture update chooses the ith update with probability $c_i(x)$ and performs the update described by P_i. With the remaining probability

$$1 - \sum_{i \in I} c_i(x)$$

the mixture update does nothing (which is the same as doing the update described by the identity kernel).

1.17.2 Radon–Nikodym Derivatives

Suppose that m is a finite signed measure and n a sigma-finite positive measure defined on the same space. We say that m is *dominated* by n or that m is *absolutely continuous with respect to* n if

$$n(A) = 0 \text{ implies } m(A) = 0, \quad \text{for all events } A.$$

We say that m is *concentrated* on a set C if

$$m(A) = m(A \cap C), \quad \text{for all events } A.$$

We say that measures m_1 and m_2 are *mutually singular* if they are concentrated on disjoint sets.

The Lebesgue–Radon–Nikodym theorem (Rudin, 1987, Theorem 6.10) says the following about m and n as defined above. Firstly, there exist unique finite signed measures m_a and m_s such that m_s and n are mutually singular, m_a is dominated by n, and $m = m_a + m_s$ (this is called the *Lebesgue decomposition*). Secondly, there exists a real-valued function f, which is unique up to redefinition on a set of n measure zero, such that

$$m_a(A) = \int_A f(x) n(dx), \quad \text{for all events } A. \tag{1.25}$$

We say that f is the *density* or *Radon–Nikodym derivative* of m with respect to n and write

$$f = \frac{dm}{dn}.$$

If n is Lebesgue measure and m is dominated by n, then f is an ordinary probability density function. If n is counting measure and m is dominated by n, then f is an ordinary probability

mass function. Hence, the Radon–Nikodym derivative generalizes these concepts. When m is not dominated by n, we have

$$\frac{dm}{dn} = \frac{dm_a}{dn}$$

so the Radon–Nikodym derivative only determines the part of m that is absolutely continuous with respect to n and says nothing about the part of m that is singular with respect to n, but that is enough for many applications.

That the Radon–Nikodym derivative f is unique only up to redefinition on a set of n measure zero would cause a problem if we made a different choice of f every time we used it, but it causes no problem if we fix one choice of f and use it always. (The same issue arises with ordinary probability density functions.)

Radon–Nikodym derivatives are often calculated using ratios. Suppose that m and n are as above and that λ is a measure that dominates both, for example, $\lambda = m + n$. Then we have

$$\frac{dm}{dn} = \frac{dm/d\lambda}{dn/d\lambda}, \tag{1.26}$$

where the right-hand side is interpreted as ordinary division when the denominator is nonzero and an arbitrary choice when the denominator is zero.

To see this, let $f_m = dm/d\lambda$ and $f_n = dn/d\lambda$, let $C = \{ x : f_n(x) \neq 0 \}$, let h be an arbitrary measurable real-valued function, and define

$$f(x) = \begin{cases} f_m(x)/f_n(x), & x \in C, \\ h(x), & x \notin C. \end{cases}$$

By the Lebesgue–Radon–Nikodym theorem, n is concentrated on C. Define a measure m_s by

$$m_s(A) = m(A \setminus C), \quad \text{for all events } A,$$

and let $m_a = m - m_s$. It remains to be shown that m_a is dominated by n and $f = dm_a/dn$. Both are shown by verifying (Equation 1.25) as follows. For any event A,

$$m_a(A) = m(A \cap C) = \int_C f_m \, d\lambda = \int_C f \cdot f_n \, d\lambda = \int_C f \, dn = \int f \, dn$$

(the last equality being that n is concentrated on C).

1.17.3 Measure-Theoretic Metropolis–Hastings

1.17.3.1 Metropolis–Hastings–Green Elementary Update

We now describe the MHG elementary update with state-dependent mixing. For $i \in I$ we have proposal mechanisms described by kernels Q_i. When the current state is x, we choose the ith proposal mechanism with probability $c_i(x)$, generating a proposal y having distribution $Q_i(x, \cdot)$.

The unnormalized measure to preserve is η (the analog of the unnormalized density h in the ordinary Metropolis–Hastings algorithm). Define measures m and m_{rev} by

$$m(B) = \iint 1_B(x, y)\eta(dx)c_i(x)Q_i(x, dy), \tag{1.27a}$$

$$m_{\text{rev}}(B) = \iint 1_B(y, x)\eta(dx)c_i(x)Q_i(x, dy), \tag{1.27b}$$

where $1_B(x, y)$ is equal to one if $(x, y) \in B$ and zero otherwise, so m and m_{rev} are measures on the Cartesian product of the sample space with itself and each B is a measurable subset of that Cartesian product. Define

$$r = \frac{dm_{\text{rev}}}{dm}. \tag{1.27c}$$

Then accept the proposal with probability $\min(1, r(x, y))$.

Note the similarity of this MHG update to the Metropolis–Hastings update (Section 1.12.1 above). It differs in the incorporation of state-dependent mixing so that $c_i(x)$ appears. It also differs in that the *Green ratio* (Equation 1.27c) is actually a Radon–Nikodym derivative rather than a simple ratio like the Hastings ratio (Equation 1.20). The "Metropolis rejection" step—accept the proposal with probability $\min(1, r)$—is the same as in the Metropolis and Metropolis–Hastings algorithms.

As we saw in Equation 1.26, a Radon–Nikodym derivative is often calculated as a ratio, so the terminology "Green ratio" for Equation 1.27c is not so strange. But our main reason for introducing this terminology is the analogy between the Metropolis ratio (Equation 1.24), the Hastings ratio (Equation 1.20), and the Green ratio (Equation 1.27c). People often write things like

$$r(x, y) = \frac{c_i(y)\eta(dy)Q_i(y, dx)}{c_i(x)\eta(dx)Q_i(x, dy)} \tag{1.28}$$

as a sloppy shorthand for actual definition via Equations 1.27a through 1.27c, but Equation 1.28 has no mathematical content other than as a mnemonic for the actual definition.

Green (1995) described a specific recipe for calculating the Green ratio (Equation 1.27c) using the ratio method (Equation 1.26) in the particular case where λ is symmetric in the sense that

$$\iint 1_B(x, y)\lambda(dx, dy) = \iint 1_B(y, x)\lambda(dx, dy) \tag{1.29}$$

for any measurable set B in the Cartesian product of the state space with itself. Such λ always exist. For example, $\lambda = m + m_{\text{rev}}$ works. Then if $f = dm/d\lambda$ and

$$C = \{\, (x, y) : f(x, y) \neq 0 \,\} \tag{1.30}$$

we have

$$r(x, y) = \begin{cases} f(y, x)/f(x, y), & x \in C, \\ 0, & x \notin C. \end{cases} \tag{1.31}$$

It does not matter whether or not we use Green's recipe for calculating (Equation 1.27c). Radon–Nikodym derivatives are unique up to redefinition on sets of measure zero, hence are the same no matter how we calculate them.

Note that the proposal distributions can be anything, described by arbitrary kernels Q_i. Thus the MHG algorithm generalizes the Metropolis–Hastings algorithm about as far as it can go. The only way your humble author can think to generalize this would be to allow state-dependent mixing over a continuum rather than a countable set of Q_i (the way state-independent mixing works; Section 1.12.8 above).

Ordinary Metropolis–Hastings samplers avoid forever the set of x such that $h(x) = 0$, where h is the unnormalized density of the equilibrium distribution (Section 1.12.1 above). Now thinking measure-theoretically, we are reminded that we may redefine h arbitrarily on sets of measure zero under the equilibrium distribution, so the set avoided depends on our choice of h. The MHG algorithm has a similar property. Suppose there is a set N that must be avoided, and $\eta(N) = 0$. Then $m_{\text{rev}}(A \times N) = 0$ for any set A, and we may choose a version of the Green ratio such that $r(x, y) = 0$ for $y \in N$. Then no proposal in N can be accepted, and the chain forever avoids N.

All MCMC ideas discussed above are special cases of the MHG algorithm. Variable-at-a-time Metropolis–Hastings updates are special cases where proposals only change one coordinate. Gibbs updates are special cases where the MHG ratio is always one and the proposal is always accepted.

1.17.3.2 The MHG Theorem

Define

$$a(x, y) = \min\bigl(1, r(x, y)\bigr),$$
$$b(x) = 1 - \int a(x, y) Q_i(x, dy).$$

The kernel describing the MHG elementary update is

$$P_i(x, A) = b(x) I(x, A) + \int_A a(x, y) Q_i(x, dy),$$

and the kernel that we must verify is reversible with respect to η is

$$K_i(x, A) = c_i(x) P_i(x, A),$$

that is, we must verify that

$$\iint g(x) h(y) \eta(dx) c_i(x) P_i(x, dy)$$

is unchanged when g and h are swapped. Since

$$\iint g(x) h(y) c_i(x) \eta(dx) P_i(x, dy) = \int g(x) h(x) b(x) c_i(x) \eta(dx) + \iint g(x) h(y) a(x, y) c_i(x) \eta(dx) Q_i(x, dy),$$

it clearly is enough to show last term is unchanged when g and h are swapped.

Suppose we have calculated the Green ratio (Equation 1.27c) using Green's recipe (Equation 1.31) with $f = dm/d\lambda$ and λ satisfying Equation 1.29. Then

$$\begin{aligned}
\iint g(x)h(y)a(x,y)c_i(x)\eta(dx)Q_i(x,dy) &= \iint g(y)h(x)a(y,x)c_i(y)\eta(dy)Q_i(y,dx) \\
&= \iint g(y)h(x)a(y,x)m_{\text{rev}}(dx,dy) \\
&= \iint_C g(y)h(x)a(y,x)m_{\text{rev}}(dx,dy) \\
&= \iint_C g(y)h(x)a(y,x)r(x,y)m(dx,dy) \\
&= \iint g(y)h(x)a(y,x)r(x,y)m(dx,dy) \\
&= \iint g(y)h(x)a(y,x)r(x,y)c_i(x)\eta(dx)Q_i(x,dy),
\end{aligned}$$

where C is defined by Equation 1.30, the first equality being the interchange of the dummy variables x and y, the second and sixth equalities being the definitions of m and m_{rev}, the third and fifth equalities being $a(y,x) = 0$ when $(x,y) \in C$, and the fourth equality being $r = dm_{\text{rev}}/dm$ and the fact that the part of m_{rev} that is dominated by m is concentrated on C, as we saw in our discussion of Equation 1.26.

Comparing the expressions at the ends of this chain of equalities, we see that it is enough to show that

$$a(y,x)r(x,y) = a(x,y), \quad \text{whenever } (x,y) \in C, \tag{1.32}$$

because the integrals are the same whether or not they are restricted to C. If $(x,y) \in C$ and $r(x,y) \le 1$, then $a(x,y) = r(x,y)$ and $a(y,x) = 1$, in which case (1.32) holds. If $(x,y) \in C$ and $1 < r(x,y)$, then $a(x,y) = 1$ and

$$a(y,x) = r(y,x) = \frac{1}{r(x,y)}$$

by Equation 1.31, in which case (Equation 1.32) holds again.

Example: Spatial Point Processes

All of this is very abstract. That is the point! But Radon–Nikodym derivatives are nothing to be frightened of. We look at some simple examples to show how the MHG algorithm works in practice.

One only needs the MHG algorithm when proposals are singular with respect to the equilibrium distribution of the Markov chain (otherwise Metropolis–Hastings would do). This often happens when the state space is the union of sets of different dimension. One example of this is spatial point processes. Geyer and Møller (1994) proposed the sampler described here independently of Green (1995), but in hindsight it is a special case of the MHG algorithm.

A spatial point process is a random pattern of points in a region A having finite measure (length, area, volume, ...), both the number of points and the positions of the points being random. A homogeneous Poisson process has a Poisson distributed number of points and the locations of the

points are independent and identically and uniformly distributed conditional on the number. We consider processes having unnormalized densities h_θ with respect to the Poisson processes.

The state space of the Poisson process is

$$\mathcal{A} = \bigcup_{n=0}^{\infty} A^n,$$

where A^0 denotes a set consisting of one point, representing the spatial pattern with no points. The probability measure of the Poisson process is defined by

$$P(B) = \sum_{n=0}^{\infty} \frac{\mu^n e^{-\mu}}{n!} \cdot \frac{\lambda^n(B \cap A^n)}{\lambda(A)^n}, \quad \text{for measurable } B \subset \mathcal{A},$$

where λ is Lebesgue measure on A and μ is an adjustable parameter (the mean number of points). To say that h_θ is an unnormalized density with respect to P means that the probability measure of the non-Poisson process is defined by

$$\begin{aligned} Q_\theta(B) &= \frac{1}{c(\theta)} \int_B h_\theta(x) P(dx) \\ &= \frac{1}{c(\theta)} \sum_{n=0}^{\infty} \frac{\mu^n e^{-\mu}}{n!} \cdot \frac{1}{\lambda(A)^n} \int_{B \cap A^n} h_\theta(x) \lambda^n(dx) \end{aligned}$$

for measurable $B \subset \mathcal{A}$, where

$$c(\theta) = \sum_{n=0}^{\infty} \frac{\mu^n e^{-\mu}}{n!} \cdot \frac{1}{\lambda(A)^n} \int h_\theta(x) \lambda^n(dx).$$

Note that the dimension of x, which is n, is different in different terms of these sums.

Let $n(x)$ denote the number of points in x. We use state-dependent mixing over a set of updates, one for each nonnegative integer i. The ith update is only valid when $n(x) = i$, in which case we propose to add one point uniformly distributed in A to the pattern, or when $n(x) = i + 1$, in which case we propose to delete a point from the pattern. (For definiteness, suppose we add or delete the last point.) The state-dependent mixing probabilities are

$$c_i(x) = \begin{cases} 1/2, & n(x) = i, \\ 1/2, & n(x) = i + 1, \\ 0, & \text{otherwise.} \end{cases}$$

For fixed x have $\sum_i c_i(x) = 1$ except when $n(x) = 0$. In that case, we do nothing (perform the identity update) with probability $1 - \sum_i c_i(x) = 1/2$ following the convention explained at the end of Section 1.17.1.

In order to apply Green's recipe for calculating Radon–Nikodym derivatives for the ith update, we need a symmetric measure on

$$(A^i \times A^{i+1}) \cup (A^{i+1} \times A^i) \tag{1.33}$$

that dominates the joint distribution m of the current state x and the proposal y or its reverse m_{rev}. This symmetric measure cannot be Lebesgue measure on Equation 1.33, because m and m_{rev} are degenerate, their first i coordinates being equal. Thus we choose the symmetric measure Λ that is the image of λ^{i+1} onto the subset of Equation 1.33 where the first i coordinates of the two parts are equal.

On the part of Equation 1.33 where $x \in A^i$ and $y \in A^{i+1}$, we have

$$f(x, y) = \frac{dm}{d\Lambda}(x, y) = \frac{\mu^i e^{-\mu} h_\theta(x)}{i! \lambda(A)^i} \cdot \frac{1}{\lambda(A)},$$

the first part on the right-hand side being the unnormalized density of the equilibrium distribution, unnormalized because we left out $c(\theta)$, which we do not know how to calculate, and the second part being the proposal density. On the part of Equation 1.33 where $x \in A^{i+1}$ and $y \in A^i$, we have

$$f(x, y) = \frac{dm}{d\Lambda}(x, y) = \frac{\mu^{i+1} e^{-\mu} h_\theta(x)}{(i+1)! \lambda(A)^{i+1}} \cdot 1,$$

the first part on the right-hand side being the unnormalized density of the equilibrium distribution, and the second part being the proposal density (which is one because deleting the last point involves no randomness). Thus the Green ratio is

$$r(x, y) = \begin{cases} \dfrac{\mu}{i+1} \cdot \dfrac{h_\theta(y)}{h_\theta(x)}, & x \in A^i \quad \text{and} \quad y \in A^{i+1}, \\ \dfrac{i+1}{\mu} \cdot \dfrac{h_\theta(y)}{h_\theta(x)}, & x \in A^{i+1} \quad \text{and} \quad y \in A^i. \end{cases}$$

We hope readers feel they could have worked this out themselves.

Since point patterns are usually considered as unordered, it is traditional to use $h_\theta(x)$ that is symmetric under exchange of points in pattern. In this case, the update that reorders the points randomly also preserves the stationary distribution. The composition of this random reordering with the update specified above (which deletes the last point) is equivalent to picking a random point to delete.

Example: Bayesian Model Selection

We consider an example done by other means in Chapter 11 of this volume. If we use MHG, there is no need for "padding" parameter vectors. We can just use the parameterization from the problem statement. If, like the ST/US sampler in Section 11.3, we only make jumps between models whose dimensions differ by one, then a very simple MHG proposal simply deletes a component of the parameter vector when moving down in dimension and adds a component distributed normally with mean zero and variance τ^2 independently of the current state when moving up in dimension. If $h(\theta)$ denotes the unnormalized posterior, then a move up in dimension from current state θ to proposed state ψ, which adds a component z to the current state, has Green ratio

$$r(\theta, \psi) = \frac{c_i(\psi) h(\psi)}{c_i(\theta) h(\theta) \phi(z/\tau)/\tau}, \tag{1.34}$$

where ϕ is the probability density function of the standard normal distribution, and a move down in dimension from current state ψ to proposed state θ, which deletes a component z from the current state, has Green ratio that is the reciprocal of the right-hand side of Equation 1.34.

1.17.4 MHG with Jacobians and Augmented State Space

Green (1995) also proposed what is in some respects a special case of MHG and in other respects an extension. We call it Metropolis–Hastings–Green with Jacobians (MHGJ). This

version is so widely used that many users think that MHGJ is the general version. This form of elementary update moves between parts of the state space that are Euclidean spaces of different dimension, hence it is often called "dimension jumping"—although that name applies to other examples, such as the preceding one, that do not involve Jacobians.

Suppose that the state space is a disjoint union

$$S = \bigcup_{m \in M} S_m,$$

where S_m is a Euclidean space of dimension d_m. We assume that the equilibrium distribution of the Markov chain is specified by an unnormalized density $h(x)$ with respect to Lebesgue measure on S. MHGJ elementary updates move from one S_m to another. Say the ith elementary update moves between $S_{m(i)}$ and $S_{n(i)}$. Thus it only makes sense to have $c_i(x) > 0$ when $x \in S_{m(i)} \cup S_{n(i)}$.

Let $U_{m(i)}$ and $U_{n(i)}$ be Euclidean spaces such that $S_{m(i)} \times U_{m(i)}$ is the same dimension as $S_{n(i)} \times U_{n(i)}$. We specify a proposal density $q_i(x, \cdot)$, which describes the conditional distribution of the proposal u given the current state x such that $u \in U_{m(i)}$ when $x \in S_{m(i)}$ and $u \in U_{n(i)}$ when $x \in S_{n(i)}$. We also specify a function g_i that maps points in $S_{m(i)} \times U_{m(i)}$ to points in $S_{n(i)} \times U_{n(i)}$ and vice versa and which is its own inverse.

The MHGJ proposal is a combination of two steps. First generate a random u from the distribution $q_i(x, \cdot)$. Then propose $g_i(x, u) = (y, v)$. The MHG ratio is

$$r(x, u, y, v) = \frac{c_i(y)h(y)q_i(y, v)}{c_i(x)h(x)q_i(x, u)} \cdot \det\bigl(\nabla g_i(x, u)\bigr),$$

the last factor being the Jacobian of the mapping g_i. This is followed by the usual Metropolis rejection: accept the proposal with probability $\min(1, r)$.

For examples of the MHGJ algorithm, see Chapter 3 (this volume).

1.17.4.1 The MHGJ Theorem

The MHGJ algorithm, because of its per-update augmentation of $U_{m(i)}$ and $U_{n(i)}$, does not exactly fit in the pattern of the MHG algorithm described above. Thus we give a separate proof.

The proof starts just like the one in Section 1.17.3.2. We see that we can deal with one arbitrary elementary update, and consequently only one pair of state augmentations. Whenever one augments the state, there are two issues: what is the equilibrium distribution on the augmented state space, and how does it relate to the distribution of interest on the original state? Here the augmented state is (x, u), the equilibrium distribution on the augmented state space has unnormalized density with respect to Lebesgue measure $h(x)q_i(x, u)$. The original state is x and the distribution of interest with unnormalized density $h(x)$ is a marginal of it. The proposal $(y, v) = g(x, u)$ is deterministic.

We now determine the Radon–Nikodym derivative of the distribution of (y, v) with respect to (x, u). We use the ratio method, determining first the Radon–Nikodym derivatives of each with respect to Lebesgue measure λ on the space where (x, u) lives. We have

$$\frac{dm}{d\lambda} = c_i(x) \cdot h(x)q_i(x, u),$$

$$\frac{dm_{\text{rev}}}{d\lambda} = c_i(y) \cdot h(y)q_i(y, v) \cdot \det\bigl(\nabla g_i(x, u)\bigr),$$

where in the latter the Jacobian arises from the multivariate change-of-variable theorem, because we are differentiating with respect to (x, u) rather than (y, v).

Acknowledgments

This chapter benefited from detailed comments by Christina Knudson, Leif Johnson, Galin Jones, and Brian Shea.

References

Bélisle, C. J. P., Romeijn, H. E., and Smith, R. L. 1993. Hit-and-run algorithms for generating multivariate distributions. *Mathematics of Operations Research*, 18:255–266.

Besag, J. and Green, P. J. 1993. Spatial statistics and Bayesian computation (with discussion). *Journal of the Royal Statistical Society, Series B*, 55:25–37.

Chan, K. S. and Geyer, C. J. 1994. Discussion of the paper by Tierney (1994). *Annals of Statistics*, 22:1747–1758.

Chen, M.-H. and Schmeiser, B. 1993. Performance of the Gibbs, hit-and-run, and Metropolis samplers. *Journal of Computational and Graphical Statistics*, 2:251–272.

Clifford, P. 1993. Discussion of Smith and Roberts (1993), Besag and Green (1993), and Gilks et al. (1993). *Journal of the Royal Statistical Society, Series B*, 55:53–54.

Freedman, D., Pisani, R., and Purves, R. 2007. *Statistics*, 4th edn. W. W. Norton, New York.

Gelfand, A. E. and Smith, A. F. M. 1990. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409.

Gelman, A., Roberts, G. O., and Gilks, W. R. 1996. Efficient Metropolis jumping rules. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith (eds), *Bayesian Statistics 5: Proceedings of the Fifth Valencia International Meeting*, pp. 599–607. Oxford University Press, Oxford.

Geman, S. and Geman, D. 1984. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Geyer, C. J. 1992. Practical Markov chain Monte Carlo (with discussion). *Statistical Science*, 7:473–511.

Geyer, C. J. 1994. On the convergence of Monte Carlo maximum likelihood calculations. *Journal of the Royal Statistical Society, Series B*, 56:261–274.

Geyer, C. J. 1999. Likelihood inference for spatial point processes. In O. E. Barndorff-Nielsen, W. S. Kendall, and M. N. M. van Lieshout (eds), *Stochastic Geometry: Likelihood and Computation*, pp. 79–140. Chapman & Hall/CRC, Boca Raton, FL.

Geyer, C. J. 2010a. Computation for the Introduction to MCMC chapter of *Handbook of Markov Chain Monte Carlo*. Technical Report 679, School of Statistics, University of Minnesota. http://purl.umn.edu/92549.

Geyer, C. J. 2010b. *mcmc: Markov Chain Monte Carlo*. R package version 0.8, available from CRAN.

Geyer, C. J. and Møller, J. 1994. Simulation and likelihood inference for spatial point processes. *Scandinavian Journal of Statistics*, 21:359–373.

Geyer, C. J. and Thompson, E. A. 1992. Constrained Monte Carlo maximum likelihood for dependent data (with discussion). *Journal of the Royal Statistical Society, Series B*, 54:657–699.

Geyer, C. J. and Thompson, E. A. 1995. Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90:909–920.

Gilks, W. R., Clayton, D. G., Spiegelhalter, D. J., Best, N. G., and McNeil, A. J. 1993. Modelling complexity: Applications of Gibbs sampling in medicine (with discussion). *Journal of the Royal Statistical Society, Series B*, 55:39–52.

Glynn, P. W. and Whitt, W. 1991. Estimating the asymptotic variance with batch means. *Operations Research Letters*, 10:431–435.

Glynn, P. W. and Whitt, W. 1992. The asymptotic validity of sequential stopping rules for stochastic simulations. *Annals of Applied Probability*, 2:180–198.

Green, P. J. 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732.

Hammersley, J. M. and Handscomb, D. C. 1964. *Monte Carlo Methods*. Methuen, London.

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

Jones, G. L. 2004. On the Markov chain central limit theorem. *Probability Surveys*, 1:299–320.

Kendall, W. S. and Møller, J. 2000. Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. *Advances in Applied Probability*, 32:844–865.

Kipnis, C. and Varadhan, S. R. S. 1986. Central limit theorem for additive functionals of reversible Markov processes and applications to simple exclusions. *Communications in Mathematical Physics*, 104:1–19.

MacEachern, S. N. and Berliner, L. M. 1994. Subsampling the Gibbs sampler. *American Statistician*, 48:188–190.

Meketon, M. S. and Schmeiser, B. W. 1984. Overlapping batch means: Something for nothing? In S. Sheppard, U. Pooch, and D. Pegden (eds), *Proceedings of the 1984 Winter Simulation Conference*, pp. 227–230. IEEE Press Piscataway, NJ.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

Meyn, S. P. and Tweedie, R. L. 1993. *Markov Chains and Stochastic Stability*. Springer, London.

Propp, J. G. and Wilson, D. B. 1996. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9:223–252.

Roberts, G. O. and Rosenthal, J. S. 1997. Geometric ergodicity and hybrid Markov chains. *Electronic Communications in Probability*, 2:13–25.

Roberts, G. O. and Rosenthal, J. S. 2004. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20–71.

Rudin, W. 1987. *Real and Complex Analysis*, 3rd edn. McGraw-Hill, New York.

Schmeiser, B. 1982. Batch size effects in the analysis of simulation output. *Operations Research*, 30:556–568.

Smith, A. F. M. and Roberts, G. O. 1993. Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods (with discussion). *Journal of the Royal Statistical Society, Series B*, 55:3–23.

Stigler, S. M. 2002. *Statistics on the Table: The History of Statistical Concepts and Methods*. Harvard University Press, Cambridge, MA.

Tanner, M. A. and Wong, W. H. 1987. The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association*, 82:528–550.

Tierney, L. 1994. Markov chains for exploring posterior distributions (with discussion). *Annals of Statistics*, 22:1701–1762.

2

A Short History of MCMC: Subjective Recollections from Incomplete Data

Christian Robert and George Casella

2.1 Introduction

Markov chain Monte Carlo (MCMC) methods have been around for almost as long as Monte Carlo techniques, even though their impact on statistics was not truly felt until the very early 1990s, except in the specialized fields of spatial statistics and image analysis, where those methods appeared earlier. The emergence of Markov based techniques in physics is a story that remains untold within this survey (see Landau and Binder, 2005). Also, we will not enter into a description of MCMC techniques, unless they have some historical link, as the remainder of this volume covers the technical aspects. A comprehensive treatment with further references can also be found in Robert and Casella (2004).

We will distinguish between the introduction of Metropolis–Hastings based algorithms and those related to Gibbs sampling, since they each stem from radically different origins, even though their mathematical justification via Markov chain theory is the same. Tracing the development of Monte Carlo methods, we will also briefly mention what we might call the "second-generation MCMC revolution." Starting in the mid to late 1990s, this includes the development of particle filters, reversible jump and perfect sampling, and concludes with more current work on population or sequential Monte Carlo and regeneration and the computing of "honest" standard errors.

As mentioned above, the realization that Markov chains could be used in a wide variety of situations only came (to mainstream statisticians) with Gelfand and Smith (1990), despite earlier publications in the statistical literature such as Hastings (1970), Geman and Geman (1984), and Tanner and Wong (1987). Several reasons can be advanced: lack of computing machinery (think of the computers of 1970!), or background on Markov chains, or hesitation to trust in the practicality of the method. It thus required visionary researchers like Gelfand and Smith to convince the community, supported by papers that demonstrated, through a series of applications, that the method was easy to understand, easy to implement and practical (Gelfand et al., 1990, 1992; Smith and Gelfand, 1992; Wakefield et al., 1994). The rapid emergence of the dedicated BUGS (Bayesian inference using Gibbs sampling) software as early as 1991, when a paper on BUGS was presented at the Valencia meeting, was another compelling argument for adopting, at large, MCMC algorithms.*

* Historically speaking, the development of BUGS can be traced back to Geman and Geman (1984) and Pearl (1987), alongside developments in the artificial intelligence community, and it pre-dates Gelfand and Smith (1990).

2.2 Before the Revolution

Monte Carlo methods were born in Los Alamos, New Mexico, during World War II, eventually resulting in the Metropolis algorithm in the early 1950s. While Monte Carlo methods were in use by that time, MCMC was brought closer to statistical practicality by the work of Hastings in the 1970s.

What can be reasonably seen as the first MCMC algorithm is what we now call the Metropolis algorithm, published by Metropolis et al. (1953). It emanates from the same group of scientists who produced the Monte Carlo method, namely the research scientists of Los Alamos, mostly physicists working on mathematical physics and the atomic bomb.

MCMC algorithms therefore date back to the same time as the development of regular (MC only) Monte Carlo methods, which are usually traced to Ulam and von Neumann in the late 1940s. Stanislaw Ulam associates the original idea with an intractable combinatorial computation he attempted in 1946 (calculating the probability of winning at the solitaire card game). This idea was enthusiastically adopted by John von Neumann for implementation with direct applications to neutron diffusion, the name "Monte Carlo" being suggested by Nicholas Metropolis. Eckhardt (1987) describes these early Monte Carlo developments, and Hitchcock (2003) gives a brief history of the Metropolis algorithm.

These occurrences very closely coincide with the appearance of the very first general-purpose digital computer, the ENIAC, which came to life in February 1946, after three years of construction. The Monte Carlo method was set up by von Neumann, who was using it on thermonuclear and fission problems as early as 1947. That same year, Ulam and von Neumann invented inversion and accept–reject techniques (also recounted in Eckhardt, 1987) to simulate from nonuniform distributions. Without computers, a rudimentary version invented by Fermi in the 1930s went unrecognized (Metropolis, 1987). Note also that, as early as 1949, a symposium on Monte Carlo was supported by Rand, the National Bureau of Standards, and the Oak Ridge laboratory and that Metropolis and Ulam (1949) published the very first paper about the Monte Carlo method.

2.2.1 The Metropolis et al. (1953) Paper

The first MCMC algorithm is associated with a second computer, called MANIAC,* built in Los Alamos under the direction of Metropolis in early 1952. Both a physicist and a mathematician, Nicholas Metropolis, came to Los Alamos in April 1943, and was to die there in 1999. The other members of the team also came to Los Alamos during those years, including the controversial Edward Teller. As early as 1942, this physicist became obsessed with the hydrogen bomb, which he eventually managed to design with Stanislaw Ulam, using the improved computer facilities of the early 1950s.

Published in June 1953 in the *Journal of Chemical Physics*, the primary focus of Metropolis et al. (1953) is the computation of integrals of the form

$$\Im = \int F(\theta) \exp\left\{\frac{-E(\theta)}{kT}\right\} \mathrm{d}\theta \Big/ \int \exp\left\{\frac{-E(\theta)}{kT}\right\} \mathrm{d}\theta,$$

* MANIAC stands for Mathematical Analyzer, Numerical Integrator and Computer.

on $\mathbb{R}^{2N}$, θ denoting a set of N particles on $\mathbb{R}^2$, with the energy E being defined as

$$E(\theta) = \frac{1}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} V(d_{ij}),$$

where V a potential function and d_{ij} the Euclidean distance between particles i and j in θ. The *Boltzmann distribution* $\exp\{-E(\theta)/kT\}$ is parameterized by the *temperature* T, k being the Boltzmann constant, with a normalization factor,

$$Z(T) = \int \exp\left\{\frac{-E(\theta)}{kT}\right\} \mathrm{d}\theta,$$

that is not available in closed form, except in trivial cases. Since θ is a $2N$-dimensional vector, numerical integration is impossible. Given the large dimension of the problem, even standard Monte Carlo techniques fail to correctly approximate $\mathfrak{I}$, since $\exp\{-E(\theta)/kT\}$ is very small for most realizations of the random configurations of the particle system (uniformly in the $2N$ square). In order to improve the efficiency of the Monte Carlo method, Metropolis et al. (1953) propose a random-walk modification of the N particles. That is, for each particle i ($1 \leq i \leq N$), values

$$x_i' = x_i + \sigma\xi_{1i} \quad \text{and} \quad y_i' = y_i + \sigma\xi_{2i}$$

are proposed, where both ξ_{1i} and ξ_{2i} are uniform $U(-1, 1)$. The energy difference ΔE between the new configuration and the previous one is then computed and the new configuration is accepted with probability

$$\min\left\{1, \exp\left(\frac{-\Delta E}{kT}\right)\right\}, \tag{2.1}$$

and otherwise the previous configuration is replicated, in the sense that its counter is increased by one in the final average of the $F(\theta_t)$s over the τ moves of the random walk ($1 \leq t \leq \tau$). Note that Metropolis et al. (1953) move one particle at a time, rather than moving all of them together, which makes the initial algorithm appear a primitive kind of Gibbs sampler!

The authors of Metropolis et al. (1953) demonstrate the validity of the algorithm by first establishing irreducibility, which they call *ergodicity*, and second proving ergodicity, that is, convergence to the stationary distribution. The second part is obtained via a discretization of the space: they first note that the proposal move is reversible, then establish that $\exp\{-E/kT\}$ is invariant. The result is therefore proven in its full generality, minus the discretization. The number of iterations of the Metropolis algorithm used in the paper seems to be limited: 16 steps for burn-in and 48–64 subsequent iterations, which required 4–5 hours on the Los Alamos computer.

An interesting variation is the *simulated annealing* algorithm, developed by Kirkpatrick et al. (1983), who connected optimization with *annealing*, the cooling of a metal. Their variation is to allow the temperature T in Equation 2.1 to decrease as the algorithm runs, according to a "cooling schedule." The simulated annealing algorithm can be shown to find the global maximum with probability 1, although the analysis is quite complex due to the fact that, with varying T, the algorithm is no longer a time-homogeneous Markov chain.

2.2.2 The Hastings (1970) Paper

The Metropolis algorithm was later generalized by Hastings (1970) and his student Peskun (1973, 1981) as a statistical simulation tool that could overcome the curse of dimensionality met by regular Monte Carlo methods, a point already emphasized in Metropolis et al. (1953).*

In his *Biometrika* paper,† Hastings (1970) also defines his methodology for finite and reversible Markov chains, treating the continuous case by using a discretization analogy. The generic probability of acceptance for a move from state i to state j is

$$\alpha_{ij} = \frac{s_{ij}}{1 + \frac{\pi_i}{\pi_j}\frac{q_{ij}}{q_{ji}}},$$

where $s_{ij} = s_{ji}$ is a positive quantity ensuring that $\alpha_{ij} \leq 1$, π_i denotes the target and q_{ij} the proposal. This generic form of probability encompasses the forms of both Metropolis et al. (1953) and Barker (1965). At this stage, Hastings says that "Little is known about the relative merits of these two choices [even though] Metropolis's method may be preferable." He also warns against "high rejection rates as indicative of a poor choice of ... transition matrix," but does not mention the opposite pitfall of low rejection rates, associated with a slow exploration of the target.

The examples in the paper include a Poisson target with a ± 1 random-walk proposal and a normal target with a uniform random-walk proposal mixed with its reflection, that is, a uniform proposal centered at $-\theta_t$ rather than at the current value θ_t of the Markov chain. On a multivariate target, Hastings introduces a Gibbs sampling strategy, updating one component at a time and defining the composed transition as satisfying the stationary condition because each component does leave the target invariant. Hastings (1970) actually refers to Erhman et al. (1960) as a preliminary, if specific, instance of this sampler. More precisely, this is Metropolis-within-Gibbs except for the name. This first introduction of the Gibbs sampler has thus been completely overlooked, even though the proof of convergence is completely general, based on a composition argument as in Tierney (1994), discussed in Section 2.4.1. The remainder of the paper deals with (a) an importance sampling version of MCMC, (b) general remarks about assessment of the error, and (c) an application to random orthogonal matrices, with another example of Gibbs sampling.

Three years later, Peskun (1973) published a comparison of Metropolis' and Barker's forms of acceptance probabilities and showed in a discrete setup that the optimal choice is that of Metropolis, where optimality is to be understood in terms of the asymptotic variance of any empirical average. The proof is a direct consequence of a result by Kemeny and Snell (1960) on the asymptotic variance. Peskun also establishes that this asymptotic variance can improve upon the independently and identically distributed (i.i.d.) case if and only if the eigenvalues of $\mathbf{P} - \mathbf{A}$ are all negative, when $\mathbf{A}$ is the transition matrix corresponding to i.i.d. simulation and $\mathbf{P}$ the transition matrix corresponding to the Metropolis algorithm, but he concludes that the trace of $\mathbf{P} - \mathbf{A}$ is always positive, therefore that the uniform improvement is impossible.

* In fact, Hastings starts by mentioning a decomposition of the target distribution into a product of one-dimensional conditional distributions, but this falls short of an early Gibbs sampler.

† Hastings (1970) is one of the ten *Biometrika* papers reproduced in Titterington and Cox (2001).

2.3 Seeds of the Revolution

A number of early pioneers had brought forward the seeds of Gibbs sampling; in particular, Hammersley and Clifford had produced a constructive argument in 1970 to recover a joint distribution from its conditionals, a result later called the *Hammersley–Clifford* theorem by Besag (1974, 1986). Besides Hastings (1970) and Geman and Geman (1984), already mentioned, other papers that contained the seeds of Gibbs sampling are Besag and Clifford (1989), Broniatowski et al. (1984), Qian and Titterington (1990), and Tanner and Wong (1987).

2.3.1 Besag and the Fundamental (Missing) Theorem

In the early 1970s, Hammersley, Clifford, and Besag were working on the specification of joint distributions from conditional distributions and on necessary and sufficient conditions for the conditional distributions to be compatible with a joint distribution. What is now known as the *Hammersley–Clifford* theorem states that a joint distribution for a vector associated with a dependence graph (edge meaning dependence and absence of edge conditional independence) must be represented as a product of functions over the *cliques* of the graphs, that is, of functions depending only on the components indexed by the labels in the clique.*

From a historical point of view, Hammersley (1974) explains why the Hammersley–Clifford theorem was never published as such, but only through Besag (1974). The reason is that Clifford and Hammersley were dissatisfied with the positivity constraint: the joint density could be recovered from the full conditionals only when the support of the joint was made up of the product of the supports of the full conditionals. While they strived *to make the theorem independent of any positivity condition*, their graduate student published a counterexample that put a full stop to their endeavors (Moussouris, 1974).

While Besag (1974) can certainly be credited to some extent with the (re)discovery of the Gibbs sampler, Besag (1975) expressed doubt about the practicality of his method, noting that "the technique is unlikely to be particularly helpful in many other than binary situations and the Markov chain itself has no practical interpretation," clearly understating the importance of his work.

A more optimistic sentiment was expressed earlier by Hammersley and Handscomb (1964), in their textbook on Monte Carlo methods. There they cover such topics as "crude Monte Carlo," importance sampling, control variates, and "conditional Monte Carlo," which looks surprisingly like a simulation approach to missing-data models (see Section 2.3.2). Of course, they do not cover the Hammersley–Clifford theorem but they do state in the Preface: "We are convinced nevertheless that Monte Carlo methods will one day reach an impressive maturity." Well said!

2.3.2 EM and Its Simulated Versions as Precursors

Due to its connection with missing-data problems, the EM algorithm (Dempster et al., 1977) has early connections with Gibbs sampling.† For instance, Broniatowski et al. (1984) and Celeux and Diebolt (1985) had tried to overcome the dependence of EM methods on the

* A clique is a maximal subset of the nodes of a graphs such that every pair of nodes within the clique is connected by an edge (Cressie, 1993).

† This is especially relevant when considering the early introduction of a Gibbs sampler by data augmentation in Tanner and Wong (1987).

starting value by replacing the E step with a *simulation* step, the missing data z_m being generated conditionally on the observation x and on the current value of the parameter θ_m. The maximization in the M step is then carried out on the simulated complete-data likelihood, $L(\theta \mid x, z_m)$, producing a new value θ_{m+1}, and this appears as a predecessor to the Gibbs step of Gelman and King (1990) and Diebolt and Robert (1994) for mixture estimation.* Unfortunately, the theoretical convergence results for these methods are limited. Celeux and Diebolt (1990) have, however, solved the convergence problem of stochastic EM (SEM) by devising a hybrid version called SAEM (for *simulated annealing EM*), where the amount of randomness in the simulations decreases with the iterations, ending up with an EM algorithm.†

2.3.3 Gibbs and Beyond

Although somewhat removed from statistical inference in the classical sense and based on earlier techniques used in statistical physics, the landmark paper by Geman and Geman (1984) brought Gibbs sampling into the arena of statistical application. This paper is also responsible for the name *Gibbs sampling*, because it implemented this method for the Bayesian study of *Gibbs random fields* which, in turn, derive their name from the physicist Josiah Willard Gibbs (1839–1903). This original implementation of the Gibbs sampler was applied to a discrete image processing problem and did not involve completion as in Section 2.3.2. But this was one more spark that led to the explosion, as it had a clear influence on Green, Smith, Spiegelhalter, and others.

The extent to which Gibbs sampling and Metropolis algorithms were in use within the image analysis and point process communities is actually quite large, as illustrated in Ripley (1987) where Section 4.7 is entitled "Metropolis' method and random fields" and describes the implementation and validation of the Metropolis algorithm in a finite setting with an application to Markov random fields and the corresponding issue of bypassing the normalizing constant. Besag et al. (1991) is another striking example of the activity in the spatial statistics community at the end of the 1980s.

2.4 The Revolution

The gap of more than 30 years between Metropolis et al. (1953) and Gelfand and Smith (1990) can still be partially attributed to the lack of appropriate computing power, as most of the examples now processed by MCMC algorithms could not have been treated previously, even though the hundreds of dimensions processed in Metropolis et al. (1953) were quite formidable. However, by the mid 1980s, the pieces were all in place.

After Peskun, MCMC in the statistical world was dormant for about 10 years, and then several papers appeared that highlighted its usefulness in specific settings such as pattern recognition, image analysis or spatial statistics. In particular, Geman and Geman (1984) influenced Gelfand and Smith (1990) to write a paper that is the genuine starting point for an intensive use of MCMC methods by the mainstream statistical community. It sparked new

* The achievement in the former paper remained unnoticed for several years due to the low-key and off-hand use of the Gibbs sampler at a time when it was unknown to most of the community.

† Other and better-known connections between EM and MCMC algorithms can be found in the literature (Liu and Rubin, 1994; Meng and Rubin, 1992; Wei and Tanner, 1990), but the connection with Gibbs sampling is more tenuous in that the simulation methods there are used to approximate quantities in a Monte Carlo fashion.

interest in Bayesian methods, statistical computing, algorithms, and stochastic processes through the use of computing algorithms such as the Gibbs sampler and the Metropolis–Hastings algorithm. Casella and George (1992) wrote an elementary introduction to the Gibbs sampler* in *The American Statistician* that disseminated the technique to a wider community while explaining in simple terms why the algorithm is valid.

Interestingly, the earlier paper by Tanner and Wong (1987) had essentially the same ingredients as Gelfand and Smith (1990), namely the fact that simulating from the conditional distributions is sufficient to asymptotically simulate from the joint. This paper was considered important enough to be a discussion paper in the *Journal of the American Statistical Association*, but its impact was somehow limited, compared with Gelfand and Smith (1990). There are several reasons for this: one is that the method seemed to apply only to missing-data problems, this impression being reinforced by the name *data augmentation*; another is that the authors were more focused on approximating the posterior distribution. They suggested an MCMC approximation to the target $\pi(\theta \mid x)$ at each iteration of the sampler, based on

$$\frac{1}{m}\sum_{k=1}^{m} \pi(\theta \mid x, z^{t,k}), \quad z^{t,k} \sim \hat{\pi}_{t-1}(z \mid x), \;\; k = 1, \ldots, m,$$

that is, by replicating m times the simulations from the current approximation $\hat{\pi}_{t-1}(z \mid x)$ of the marginal posterior distribution of the missing data. This focus on estimation of the posterior distribution connected the original data augmentation algorithm to EM, as pointed out by Dempster in the discussion. Although the discussion by Morris gets very close to the two-stage Gibbs sampler for hierarchical models, he is still concerned about doing m iterations, and worries about how costly that would be. Tanner and Wong mention taking $m = 1$ at the end of the paper, referring to this as an "extreme case."

In a sense, Tanner and Wong (1987) was still too close to Rubin's (1978) multiple imputation to start a new revolution. Yet another reason for this may be that the theoretical background was based on functional analysis rather than Markov chain theory, which needed, in particular, the Markov kernel to be uniformly bounded and equicontinuous. This may have discouraged potential users as requiring too much mathematics.

The authors of this review were fortunate enough to attend many focused conferences during this time, where we were able to witness the explosion of Gibbs sampling. In the summer of 1986 in Bowling Green, Ohio, Smith gave a series of ten lectures on hierarchical models. Although there was a lot of computing mentioned, the Gibbs sampler was not yet fully developed. (Interestingly, Smith commented that the limiting factor, at that time, for the full exploitation of hierarchical models in statistical problems was the inability to compute high-dimensional integrals.) In another lecture in June 1989 at a Bayesian workshop in Sherbrooke, Quebec, he revealed for the first time the generic features of Gibbs sampler, and we still remember vividly the shock induced in ourselves and in the whole audience by the sheer breadth of the method: this development of Gibbs sampling, MCMC, and the resulting seminal paper of Gelfand and Smith (1990) was an *epiphany*† in the world of statistics.

* On a humorous note, the original Technical Report of this paper was called *Gibbs for Kids*, which was changed because a referee did not appreciate the humor. However, our colleague Dan Gianola, an animal breeder at Wisconsin, liked the title. In using Gibbs sampling in his work, he gave a presentation in 1993 at the 44th Annual Meeting of the European Association for Animal Production, Aarhus, Denmark. The title: *Gibbs for Pigs*.

† Epiphany, *n.* A spiritual event in which the essence of a given object of manifestation appears to the subject, as in a sudden flash of recognition.

The explosion had begun, and just two years later an MCMC conference at Ohio State University, organized by Gelfand, Goel, and Smith, consisted of three full days of talks. Many of the talks were to become influential papers; including Albert and Chib (1993), Gelman and Rubin (1992), Geyer (1992), Gilks (1992), Liu et al. (1994, 1995), and Tierney (1994).

Approximately one year later, in May 1992, there was a meeting of the Royal Statistical Society on "The Gibbs sampler and other Markov chain Monte Carlo methods," where four papers were presented followed by much discussion. The papers appear in the first issue of the *Journal of the Royal Statistical Society*, *Series B*, in 1993, together with 49 (!) pages of discussion. The excitement is clearly evident in the writings, even though the theory and implementation were not always perfectly understood.

Looking at these meetings, we can see the paths that Gibbs sampling would lead us down. In the next two sections we will summarize some of the advances from the early to mid 1990s.

2.4.1 Advances in MCMC Theory

Perhaps the most influential MCMC theory paper of the 1990s is Tierney (1994), which carefully laid out all of the assumptions needed to analyze the Markov chains and then developed their properties, in particular, convergence of ergodic averages and central limit theorems. In one of the discussions of that paper, Chan and Geyer (1994) were able to relax a condition on Tierney's central limit theorem, and this new condition plays an important role in research today (see Section 2.5.4). A pair of very influential, and innovative, papers is the work of Liu et al. (1994, 1995), who very carefully analyzed the covariance structure of Gibbs sampling, and were able to formally establish the validity of Rao-Blackwellization in Gibbs sampling. Gelfand and Smith (1990) had used Rao-Blackwellization, but it was not justified at that time, as the original theorem was only applicable to i.i.d. sampling, which is not the case in MCMC. Another significant entry is Rosenthal (1995), who obtained one of the earliest results on exact rates of convergence.

Another paper must be singled out, namely Mengersen and Tweedie (1996), for setting the tone for the study of the speed of convergence of MCMC algorithms to the target distribution. Subsequent works in this area by Richard Tweedie, Gareth Roberts, Jeff Rosenthal and co-authors are too numerous to be mentioned here, although the paper by Roberts et al. (1997) must be cited for setting explicit targets on the acceptance rate of the random-walk Metropolis–Hastings algorithm, as well as Roberts and Rosenthal (1999) for obtaining an upper bound on the number of iterations (523) needed to approximate the target up to 1% by a slice sampler. The untimely death of Richard Tweedie in 2001 also had a major impact on the book about MCMC convergence he was contemplating with Gareth Roberts.

One pitfall arising from the widespread use of Gibbs sampling was the tendency to specify models only through their conditional distributions, almost always without referring to the positivity conditions in Section 2.3. Unfortunately, it is possible to specify a perfectly legitimate-looking set of conditionals that do not correspond to any joint distribution, and the resulting Gibbs chain cannot converge. Hobert and Casella (1996) were able to document the conditions needed for a convergent Gibbs chain, and alerted the Gibbs community to this problem, which only arises when improper priors are used, but this is a frequent occurrence.

Much other work followed, and continues to grow today. Geyer and Thompson (1995) describe how to put a "ladder" of chains together for both "hot" and "cold" exploration, followed by Neal's (1996) introduction of tempering; Athreya et al. (1996) gave more easily

verifiable conditions for convergence; Meng and van Dyk (1999) and Liu and Wu (1999) developed the theory of parameter expansion in the data augmentation algorithm, leading to construction of chains with faster convergence, and to the work of Hobert and Marchev (2008), who give precise constructions and theorems to show how parameter expansion can uniformly improve over the original chain.

2.4.2 Advances in MCMC Applications

The real reason for the explosion of MCMC methods was the fact that an enormous number of problems that were deemed to be computational nightmares now cracked open like eggs. As an example, consider this very simple random effects model from Gelfand and Smith (1990). Observe

$$Y_{ij} = \theta_i + \varepsilon_{ij}, \quad i = 1, \ldots, K, \;\; j = 1, \ldots, J, \tag{2.2}$$

where

$$\begin{aligned} \theta_i &\sim N(\mu, \sigma_\theta^2) \\ \varepsilon_{ij} &\sim N(0, \sigma_\varepsilon^2), \quad \text{independent of } \theta_i. \end{aligned}$$

Estimation of the variance components can be difficult for a frequentist (restricted maximum likelihood is typically preferred) but it was a nightmare for a Bayesian, as the integrals were intractable. However, with the usual priors on μ, σ_θ^2, and σ_ε^2, the full conditionals are trivial to sample from and the problem is easily solved via Gibbs sampling. Moreover, we can increase the number of variance components and the Gibbs solution remains easy to implement.

During the early 1990s, researchers found that Gibbs, or Metropolis–Hastings, algorithms would crack almost any problem that they looked at, and there was a veritable flood of papers applying MCMC to previously intractable models and getting good solutions. For example, building on Equation 2.2, it was quickly realized that Gibbs sampling was an easy route to getting estimates in the linear mixed models (Wang et al., 1993, 1994), and even generalized linear mixed models (Zeger and Karim, 1991). Building on the experience gained with the EM algorithm, similar arguments made it possible to analyze probit models using a latent variable approach in a linear mixed model (Albert and Chib, 1993) and in mixture models with Gibbs sampling (Diebolt and Robert, 1994). It progressively dawned on the community that latent variables could be artificially introduced to run the Gibbs sampler in just about every situation, as eventually published in Damien et al. (1999), the main example being the slice sampler (Neal, 2003). A very incomplete list of some other applications includes change-point analysis (Carlin et al., 1992; Stephens, 1994), genomics (Churchill, 1995; Lawrence et al., 1993; Stephens and Smith, 1993), capture–recapture (Dupuis, 1995; George and Robert, 1992), variable selection in regression (George and McCulloch, 1993), spatial statistics (Raftery and Banfield, 1991), and longitudinal studies (Lange et al., 1992).

Many of these applications were advanced though other developments such as the adaptive rejection sampling of Gilks (1992) and Gilks et al. (1995), and the simulated tempering approaches of Geyer and Thompson (1995) or Neal (1996).

2.5 After the Revolution

After the revolution comes the "second" revolution, but now we have a more mature field. The revolution has slowed, and the problems are being solved in, perhaps, deeper and more sophisticated ways, even though Gibbs sampling also offers the amateur the possibility of handling Bayesian analysis in complex models at little cost, as exhibited by the widespread use of BUGS, which mostly focuses on this approach.* But, as before, the methodology continues to expand the set of problems for which statisticians can provide meaningful solutions, and thus continues to further the impact of statistics.

2.5.1 A Brief Glimpse at Particle Systems

The realization of the possibilities of iterating importance sampling is not new: in fact, it is about as old as Monte Carlo methods themselves. It can be found in the molecular simulation literature of the 1950s, for example Hammersley and Morton (1954), Rosenbluth and Rosenbluth (1955), and Marshall (1965). Hammersley and colleagues proposed such a method to simulate a self-avoiding random walk (see Madras and Slade, 1993) on a grid, due to huge inefficiencies in regular importance sampling and rejection techniques. Although this early implementation occurred in particle physics, the use of the term "particle" only dates back to Kitagawa (1996), while Carpenter et al. (1997) coined the term "particle filter." In signal processing, early occurrences of a particle filter can be traced back to Handschin and Mayne (1969).

More in connection with our theme, the landmark paper of Gordon et al. (1993) introduced the bootstrap filter which, while formally connected with importance sampling, involves past simulations and possible MCMC steps (Gilks and Berzuini, 2001). As described in the volume edited by Doucet et al. (2001), particle filters are simulation methods adapted to sequential settings where data are collected progressively in time, as in radar detection, telecommunication correction or financial volatility estimation. Taking advantage of state–space representations of those dynamic models, particle filter methods produce Monte Carlo approximations to the posterior distributions by propagating simulated samples whose weights are actualized against the incoming observations. Since the importance weights have a tendency to degenerate, that is, all weights but one are close to zero, additional MCMC steps can be introduced at times to recover the variety and representativeness of the sample. Modern connections with MCMC in the construction of the proposal kernel are to be found, for instance, in Doucet et al. (2000) and Del Moral et al. (2006). In parallel, sequential imputation was developed in Kong et al. (1994), while Liu and Chen (1995) first formally pointed out the importance of resampling in sequential Monte Carlo, a term coined by them.

The recent literature on the topic more closely bridges the gap between sequential Monte Carlo and MCMC methods by making adaptive MCMC a possibility—see, for example, Andrieu et al. (2004) or Roberts and Rosenthal (2005).

2.5.2 Perfect Sampling

Introduced in the seminal paper of Propp and Wilson (1996), perfect sampling, namely the ability to use MCMC methods to produce an exact (or perfect) simulation from the target,

* BUGS now uses both Gibbs sampling and Metropolis–Hastings algorithms.

has a unique place in the history of MCMC methods. Although this exciting discovery led to an outburst of papers, in particular in the large body of work of Møller and coauthors, including the book by Møller and Waagepetersen (2004), as well as many reviews and introductory materials, such as Casella et al. (2001), Fismen (1998), and Dimakos (2001), the excitement quickly died down. The major reason for this ephemeral lifespan is that the construction of perfect samplers is most often close to impossible or impractical, despite some advances in implementation (Fill, 1998a,b).

There is, however, ongoing activity in the area of point processes and stochastic geometry, much from the work of Møller and Kendall. In particular, Kendall and Møller (2000) developed an alternative to the *coupling from the past* (CFTP) algorithm of Propp and Wilson (1996), called *horizontal CFTP*, which mainly applies to point processes and is based on continuous-time birth-and-death processes. See also Fernández et al. (1999) for another horizontal CFTP algorithm for point processes. Berthelsen and Møller (2003) exhibited a use of these algorithms for nonparametric Bayesian inference on point processes.

2.5.3 Reversible Jump and Variable Dimensions

From many viewpoints, the invention of the reversible jump algorithm in Green (1995) can be seen as the start of the second MCMC revolution: the formalization of a Markov chain that moves across models and parameter spaces allowed for the Bayesian processing of a wide variety of new models and contributed to the success of Bayesian model choice and subsequently to its adoption in other fields. There exist earlier alternative Monte Carlo solutions such as Gelfand and Dey (1994) and Carlin and Chib (1995), the latter being very close in spirit to reversible jump MCMC (as shown by the completion scheme of Brooks et al., 2003), but the definition of a proper balance condition on cross-model Markov kernels in Green (1995) gives a generic setup for exploring variable dimension spaces, even when the number of models under comparison is infinite. The impact of this new idea was clearly perceived when looking at the First European Conference on highly structured stochastic systems that took place in Rebild, Denmark, the next year, organized by Stephen Lauritzen and Jesper Møller: a large majority of the talks were aimed at direct implementations of RJMCMC to various inference problems. The application of RJMCMC to mixture order estimation in the discussion paper of Richardson and Green (1997) ensured further dissemination of the technique. More recently, Stephens (2000) proposed a continuous-time version of RJMCMC, based on earlier ideas of Geyer and Møller (1994), but with similar properties (Cappé et al., 2003), while Brooks et al. (2003) made proposals for increasing the efficiency of the moves. In retrospect, while reversible jump is somehow unavoidable in the processing of very large numbers of models under comparison, as for instance in variable selection (Marin and Robert, 2007), the implementation of a complex algorithm such as RJMCMC for the comparison of a few models is somewhat of an overkill since there exist alternative solutions based on model-specific MCMC chains (e.g. Chen et al., 2000).

2.5.4 Regeneration and the Central Limit Theorem

While the central limit theorem (CLT) is a central tool in Monte Carlo convergence assessment, its use in MCMC setups took longer to emerge, despite early signals by Geyer (1992), and it is only recently that sufficiently clear conditions emerged. We recall that the ergodic theorem (see, e.g. Robert and Casella, 2004, Theorem 6.63) states that, if $(\theta_t)_t$ is a Markov chain with stationary distribution π, and $h(\cdot)$ is a function with finite variance, then under

fairly mild conditions,

$$\lim_{n\to\infty} \bar{h}_n = \int h(\theta)\pi(\theta)\, d\theta = \mathrm{E}_\pi[h(\theta)],$$

almost everywhere, where $\bar{h}_n = (1/n)\sum_{i=1}^{n} h(\theta_i)$. For the CLT to be used to monitor this convergence,

$$\frac{\sqrt{n}(\bar{h}_n - \mathrm{E}_\pi[h(\theta)])}{\sqrt{\mathrm{var}[h(\theta)]}} \to N(0,1), \tag{2.3}$$

there are two roadblocks. First, convergence to normality is strongly affected by the lack of independence. To get CLTs for Markov chains, we can use a result of Kipnis and Varadhan (1986), which requires the chain to be reversible, as is the case for Metropolis–Hastings chains, or we must delve into mixing conditions (Billingsley, 1995, Section 27), which are typically not easy to verify. However, Chan and Geyer (1994) showed how the condition of geometric ergodicity could be used to establish CLTs for Markov chains. But getting the convergence is only half of the problem. In order to use Equation 2.3, we must be able to consistently estimate the variance, which turns out to be another difficult endeavor. The "naive" estimate of the usual standard error is not consistent in the dependent case and the most promising paths for consistent variance estimates seem to be through regeneration and batch means.

The theory of regeneration uses the concept of a split chain (Athreya and Ney, 1978), and allows us to independently restart the chain while preserving the stationary distribution. These independent "tours" then allow the calculation of consistent variance estimates and honest monitoring of convergence through Equation 2.3. Early work on applying regeneration to MCMC chains was done by Mykland et al. (1995) and Robert (1995), who showed how to construct the chains and use them for variance calculations and diagnostics (see also Guihenneuc-Jouyaux and Robert, 1998), as well as deriving adaptive MCMC algorithms (Gilks et al., 1998). Rosenthal (1995) also showed how to construct and use regenerative chains, and much of this work is reviewed in Jones and Hobert (2001). The most interesting and practical developments, however, are in Hobert et al. (2002) and Jones et al. (2006), where consistent estimators are constructed for var$[h(\theta)]$, allowing valid monitoring of convergence in chains that satisfy the CLT. Interestingly, although Hobert et al. (2002) uses regeneration, Jones et al. (2006) get their consistent estimators thorough another technique, that of consistent batch means.

2.6 Conclusion

The impact of Gibbs sampling and MCMC on Bayesian statistics was to change our entire method of thinking about and attacking problems, representing a *paradigm shift* (Kuhn, 1996). Now, the collection of real problems that we could solve grew almost without bound. Markov chain Monte Carlo changed our emphasis from "closed form" solutions to algorithms, expanded our impact to solving "real" applied problems and to improving numerical algorithms using statistical ideas, and led us into a world where "exact" now means "simulated."

This has truly been a quantum leap in the evolution of the field of statistics, and the evidence is that there are no signs of a slowdown. Although the "explosion" is over, the

current work is going deeper into theory and applications, and continues to expand our horizons and influence by increasing our ability to solve even bigger and more important problems. The size of the data sets, and of the models, for example in genomics or climatology, is something that could not have been conceived 60 years ago, when Ulam and von Neumann invented the Monte Carlo method. Now we continue to plod on, and hope that the advances that we make here will, in some way, help our colleagues 60 years in the future solve problems that we cannot yet conceive.

Acknowledgments

Christian Robert was supported by the Agence Nationale de la Recherche, Paris, through the 2006–2008 project ANR=05-BLAN-0299 Adap'MC. This work was partly done during a visit to the Queensland University of Technology, Brisbane, and CR is grateful to Kerrie Mengersen for her hospitality and support. George Casella was supported by National Science Foundation Grants DMS-04-05543, DMS-0631632 and SES-0631588.

We are grateful for comments and suggestions from Olivier Cappé, Alan Gelfand, Peter Green, Jun Liu, Sharon McGrayne, Peter Müller, Gareth Roberts, Adrian Smith, and David Spiegelhalter.

References

Albert, J. and Chib, S. 1993. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88:669–679.

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. 2004. An introduction to MCMC for machine learning. *Machine Learning*, 50(1–2):5–43.

Athreya, K., Doss, H., and Sethuraman, J. 1996. On the convergence of the Markov chain simulation method. *Annals of Statistics*, 24:69–100.

Athreya, K. and Ney, P. 1978. A new approach to the limit theory of recurrent Markov chains. *Transactions of the American Mathematical Society*, 245:493–501.

Barker, A. 1965. Monte Carlo calculations of the radial distribution functions for a proton–electron plasma. *Australian Journal of Physics*, 18:119–133.

Berthelsen, K. and Møller, J. 2003. Likelihood and non-parametric Bayesian MCMC inference for spatial point processes based on perfect simulation and path sampling. *Scandinavian Journal of Statistics*, 30:549–564.

Besag, J. 1974. Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal of the Royal Statistical Society, Series B*, 36:192–326.

Besag, J. 1975. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195.

Besag, J. 1986. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, 48:259–279.

Besag, J. and Clifford, P. 1989. Generalized Monte Carlo significance tests. *Biometrika*, 76:633–642.

Besag, J., York, J., and Mollie, A. 1991. Bayesian image restoration, with two applications in spatial statistics (with discussion). *Annals of the Institute of Statistical Mathematics*, 42(1):1–59.

Billingsley, P. 1995. *Probability and Measure*, 3rd edn. Wiley, New York.

Broniatowski, M., Celeux, G., and Diebolt, J. 1984. Reconnaissance de mélanges de densités par un algorithme d'apprentissage probabiliste. In E. Diday (ed.), *Data Analysis and Informatics*, Volume 3, pp. 359–373. North-Holland, Amsterdam.

Brooks, S. P., Giudici, P., and Roberts, G. O. 2003. Efficient construction of reversible jump Markov chain Monte Carlo proposal distributions (with discussion). *Journal of the Royal Statistical Society, Series B*, 65(1):3–55.

Cappé, O., Robert, C., and Rydén, T. 2003. Reversible jump, birth-and-death, and more general continuous time MCMC samplers. *Journal of the Royal Statistical Society, Series B*, 65(3): 679–700.

Carlin, B. and Chib, S. 1995. Bayesian model choice through Markov chain Monte Carlo. *Journal of the Royal Statistical Society, Series B*, 57(3):473–484.

Carlin, B., Gelfand, A., and Smith, A. 1992. Hierarchical Bayesian analysis of change point problems. *Applied Statistics*, 41:389–405.

Carpenter, J., Clifford, P., and Fernhead, P. 1997. Building robust simulation-based filters for evolving datasets. Technical report, Department of Statistics, Oxford University.

Casella, G. and George, E. 1992. An introduction to Gibbs sampling. *American Statistician*, 46:167–174.

Casella, G., Lavine, M., and Robert, C. 2001. Explaining the perfect sampler. *American Statistician*, 55(4):299–305.

Celeux, G. and Diebolt, J. 1985. The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Computational Statistics Quaterly*, 2:73–82.

Celeux, G. and Diebolt, J. 1990. Une version de type recuit simulé de l'algorithme EM. *Comptes Rendus de l'Académie des Sciences (Paris)*, 310:119–124.

Chan, K. and Geyer, C. 1994. Discussion of "Markov chains for exploring posterior distribution." *Annals of Statistics*, 22:1747–1758.

Chen, M., Shao, Q., and Ibrahim, J. 2000. *Monte Carlo Methods in Bayesian Computation*. Springer, New York.

Churchill, G. 1995. Accurate restoration of DNA sequences (with discussion). In C. Gatsonis, J. S. Hodges, R. Kass, and N. Singpurwalla (eds), *Case Studies in Bayesian Statistics*, Volume 2, pp. 90–148. Springer, New York.

Cressie, N. 1993. *Spatial Statistics*. Wiley, New York.

Damien, P., Wakefield, J., and Walker, S. 1999. Gibbs sampling for Bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society, Series B*, 61(2):331–344.

Del Moral, P., Doucet, A., and Jasra, A. 2006. The sequential Monte Carlo samplers. *Journal of the Royal Statistical Society, Series B*, 68(3):411–436.

Dempster, A., Laird, N., and Rubin, D. 1977. Maximum likelihood from incomplete data via the *EM* algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, 39:1–38.

Diebolt, J. and Robert, C. 1994. Estimation of finite mixture distributions by Bayesian sampling. *Journal of the Royal Statistical Society, Series B*, 56:363–375.

Dimakos, X. K. 2001. A guide to exact simulation. *International Statistical Review*, 69(1):27–48.

Doucet, A., de Freitas, N., and Gordon, N. 2001. *Sequential Monte Carlo Methods in Practice*. Springer, New York.

Doucet, A., Godsill, S., and Andrieu, C. 2000. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208.

Dupuis, J. 1995. Bayesian estimation of movement probabilities in open populations using hidden Markov chains. *Biometrika*, 82(4):761–772.

Eckhardt, R. 1987. Stan Ulam, John von Neumann, and the Monte Carlo method. *Los Alamos Science, Special Issue*, pp. 131–141. Available at http://library.lanl.gov.cgi–bin/getfile?15–13.pdf.

Erhman, J., Fosdick, L., and Handscomb, D. 1960. Computation of order parameters in an Ising lattice by the Monte Carlo method. *Journal of Mathematical Physics*, 1:547–558.

Fernández, R., Ferrari, P., and Garcia, N. L. 1999. Perfect simulation for interacting point processes, loss networks and Ising models. Technical report, Laboratoire Raphael Salem, Universitè de Rouen.

Fill, J. 1998a. An interruptible algorithm for exact sampling via Markov chains. *Annals of Applied Probability*, 8:131–162.

Fill, J. 1998b. The move-to front rule: A case study for two perfect sampling algorithms. *Probability in the Engineering and Informational Sciences*, 8:131–162.

Fismen, M. 1998. Exact simulation using Markov chains. Technical Report 6/98, Institutt for Matematiske Fag, Oslo. Diploma thesis.

Gelfand, A. and Dey, D. 1994. Bayesian model choice: asymptotics and exact calculations. *Journal of the Royal Statistical Society, Series B*, 56:501–514.

Gelfand, A., Hills, S., Racine-Poon, A., and Smith, A. 1990. Illustration of Bayesian inference in normal data models using Gibbs sampling. *Journal of the American Statistical Association*, 85:972–982.

Gelfand, A. and Smith, A. 1990. Sampling based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409.

Gelfand, A., Smith, A., and Lee, T. 1992. Bayesian analysis of constrained parameters and truncated data problems using Gibbs sampling. *Journal of the American Statistical Association*, 87: 523–532.

Gelman, A. and King, G. 1990. Estimating the electoral consequences of legislative redistricting. *Journal of the American Statistical Association*, 85:274–282.

Gelman, A. and Rubin, D. 1992. Inference from iterative simulation using multiple sequences (with discussion). *Statistical Science*, 7:457–511.

Geman, S. and Geman, D. 1984. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

George, E. and McCulloch, R. 1993. Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88:881–889.

George, E. and Robert, C. 1992. Calculating Bayes estimates for capture-recapture models. *Biometrika*, 79(4):677–683.

Geyer, C. 1992. Practical Monte Carlo Markov chain (with discussion). *Statistical Science*, 7:473–511.

Geyer, C. and Møller, J. 1994. Simulation procedures and likelihood inference for spatial point processes. *Scandinavian Journal of Statistics*, 21:359–373.

Geyer, C. and Thompson, E. 1995. Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90:909–920.

Gilks, W. 1992. Derivative-free adaptive rejection sampling for Gibbs sampling. In J. Bernardo, J. Berger, A. Dawid, and A. Smith (eds), *Bayesian Statistics 4*, pp. 641–649. Oxford University Press, Oxford.

Gilks, W. and Berzuini, C. 2001. Following a moving target—Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society, Series B*, 63(1):127–146.

Gilks, W., Best, N., and Tan, K. 1995. Adaptive rejection Metropolis sampling within Gibbs sampling. *Applied Statistics*, 44:455–472.

Gilks, W., Roberts, G. O., and Sahu, S. 1998. Adaptive Markov chain Monte Carlo. *Journal of the American Statistical Association*, 93:1045–1054.

Gordon, N., Salmond, J., and Smith, A. 1993. A novel approach to non-linear/non-Gaussian Bayesian state estimation. *IEEE Proceedings on Radar and Signal Processing*, 140:107–113.

Green, P. 1995. Reversible jump MCMC computation and Bayesian model determination. *Biometrika*, 82(4):711–732.

Guihenneuc-Jouyaux, C. and Robert, C. 1998. Finite Markov chain convergence results and MCMC convergence assessments. *Journal of the American Statistical Association*, 93:1055–1067.

Hammersley, J. 1974. Discussion of Mr Besag's paper. *Journal of the Royal Statistical Society, Series B*, 36:230–231.

Hammersley, J. and Handscomb, D. 1964. *Monte Carlo Methods*. Wiley, New York.

Hammersley, J. and Morton, K. 1954. Poor man's Monte Carlo. *Journal of the Royal Statistical Society, Series B*, 16:23–38.

Handschin, J. and Mayne, D. 1969. Monte Carlo techniques to estimate the conditional expectation in multi-stage nonlinear filtering. *International Journal of Control*, 9:547–559.

Hastings, W. 1970. Monte Carlo sampling methods using Markov chains and their application. *Biometrika*, 57:97–109.

Hitchcock, D. B. 2003. A history of the Metropolis-Hastings algorithm. *American Statistician*, 57(4): 254–257.

Hobert, J. and Casella, G. 1996. The effect of improper priors on Gibbs sampling in hierarchical linear models. *Journal of the American Statistical Association*, 91:1461–1473.

Hobert, J., Jones, G., Presnell, B., and Rosenthal, J. 2002. On the applicability of regenerative simulation in Markov chain Monte Carlo. *Biometrika*, 89(4):731–743.

Hobert, J. and Marchev, D. 2008. A theoretical comparison of the data augmentation, marginal augmentation and PX-DA algorithms. *Annals of Statistics*, 36:532–554.

Jones, G., Haran, M., Caffo, B. S., and Neath, R. 2006. Fixed-width output analysis for Markov chain Monte Carlo. *Journal of the American Statistical Association*, 101:1537–1547.

Jones, G. and Hobert, J. 2001. Honest exploration of intractable probability distributions via Markov chain Monte Carlo. *Statistical Science*, 16(4):312–334.

Kemeny, J. and Snell, J. 1960. *Finite Markov Chains*. Van Nostrand, Princeton, NJ.

Kendall, W. and Møller, J. 2000. Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. *Advances in Applied Probability*, 32: 844–865.

Kipnis, C. and Varadhan, S. 1986. Central limit theorem for additive functionals of reversible Markov processes and applications to simple exclusions. *Communications in Mathematical Physics*, 104: 1–19.

Kirkpatrick, S., Gelatt, C., and Vecchi, M. 1983. Optimization by simulated annealing. *Science*, 220: 671–680.

Kitagawa, G. 1996. Monte Carlo filter and smoother for non-Gaussian non-linear state space models. *Journal of Computational and Graphical Statistics*, 5:1–25.

Kong, A., Liu, J., and Wong, W. 1994. Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89:278–288.

Kuhn, T. 1996. *The Structure of scientific Revolutions*, 3rd edn. University of Chicago Press, Chicago.

Landau, D. and Binder, K. 2005. *A Guide to Monte Carlo Simulations in Statistical Physics*, 2nd edn. Cambridge University Press, Cambridge.

Lange, N., Carlin, B. P., and Gelfand, A. E. 1992. Hierarchal Bayes models for the progression of hiv infection using longitudinal cd4 t-cell numbers. *Journal of the American Statistical Association*, 87:615–626.

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. 1993. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214.

Liu, C. and Rubin, D. 1994. The ECME algorithm: a simple extension of EM and ECM with faster monotonous convergence. *Biometrika*, 81:633–648.

Liu, J. and Chen, R. 1995. Blind deconvolution via sequential imputations. *Journal of the American Statistical Association*, 90:567–576.

Liu, J., Wong, W., and Kong, A. 1994. Covariance structure of the Gibbs sampler with applications to the comparisons of estimators and sampling schemes. *Biometrika*, 81:27–40.

Liu, J., Wong, W., and Kong, A. 1995. Correlation structure and convergence rate of the Gibbs sampler with various scans. *Journal of the Royal Statistical Society, Series B*, 57:157–169.

Liu, J. and Wu, Y. N. 1999. Parameter expansion for data augmentation. *Journal of the American Statistical Association*, 94:1264–1274.

Madras, N. and Slade, G. 1993. *The Self-Avoiding Random Walk*. Birkhäuser, Boston.

Marin, J.-M. and Robert, C. 2007. *Bayesian Core*. Springer, New York.

Marshall, A. 1965. The use of multi-stage sampling schemes in Monte Carlo computations. In *Symposium on Monte Carlo Methods*. Wiley, New York.

Meng, X. and Rubin, D. 1992. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80:267–278.

Meng, X. and van Dyk, D. 1999. Seeking efficient data augmentation schemes via conditional and marginal augmentation. *Biometrika*, 86:301–320.
Mengersen, K. and Tweedie, R. 1996. Rates of convergence of the Hastings and Metropolis algorithms. *Annals of Statistics*, 24:101–121.
Metropolis, N. 1987. The beginning of the Monte Carlo method. *Los Alamos Science*, 15:125–130.
Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092.
Metropolis, N. and Ulam, S. 1949. The Monte Carlo method. *Journal of the American Statistical Association*, 44:335–341.
Møller, J. and Waagepetersen, R. 2004. *Statistical Inference and Simulation for Spatial Point Processes*. Chapman and Hall/CRC, Boca Raton, FL.
Moussouris, J. 1974. Gibbs and Markov random systems with constraints. *Journal of Statistical Physics*, 10:11–33.
Mykland, P., Tierney, L., and Yu, B. 1995. Regeneration in Markov chain samplers. *Journal of the American Statistical Association*, 90:233–241.
Neal, R. 1996. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6:353–356.
Neal, R. 2003. Slice sampling (with discussion). *Annals of Statistics*, 31:705–767.
Pearl, J. 1987. Evidential reasoning using stochastic simulation in causal models. *Artificial Intelligence*, 32:247–257.
Peskun, P. 1973. Optimum Monte Carlo sampling using Markov chains. *Biometrika*, 60:607–612.
Peskun, P. 1981. Guidelines for chosing the transition matrix in Monte Carlo methods using Markov chains. *Journal of Computational Physics*, 40:327–344.
Propp, J. and Wilson, D. 1996. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9:223–252.
Qian, W. and Titterington, D. 1990. Parameter estimation for hidden Gibbs chains. *Statistics & Probability Letters*, 10:49–58.
Raftery, A. and Banfield, J. 1991. Stopping the Gibbs sampler, the use of morphology, and other issues in spatial statistics. *Annals of the Institute of Statistical Mathematics*, 43:32–43.
Richardson, S. and Green, P. 1997. On Bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society, Series B*, 59:731–792.
Ripley, B. 1987. *Stochastic Simulation*. Wiley, New York.
Robert, C. 1995. Convergence control techniques for MCMC algorithms. *Statistical Science*, 10(3): 231–253.
Robert, C. and Casella, G. 2004. *Monte Carlo Statistical Methods*, 2nd edn. Springer, New York.
Roberts, G. O., Gelman, A., and Gilks, W. 1997. Weak convergence and optimal scaling of random walk Metropolis algorithms. *Annals of Applied Probability*, 7:110–120.
Roberts, G. O. and Rosenthal, J. 1999. Convergence of slice sampler Markov chains. *Journal of the Royal Statistical Society, Series B*, 61:643–660.
Roberts, G. O. and Rosenthal, J. 2005. Coupling and ergodicity of adaptive mcmc. *Journal of Applied Probability*, 44:458–475.
Rosenbluth, M. and Rosenbluth, A. 1955. Monte Carlo calculation of the average extension of molecular chains. *Journal of Chemical Physics*, 23:356–359.
Rosenthal, J. 1995. Minorization conditions and convergence rates for Markov chain Monte Carlo. *Journal of the American Statistical Association*, 90:558–566.
Rubin, D. 1978. Multiple imputation in sample surveys: a phenomenological Bayesian approach to nonresponse. In *Imputation and Editing of Faulty or Missing Survey Data*. Washington, DC: US Department of Commerce.
Smith, A. and Gelfand, A. 1992. Bayesian statistics without tears: A sampling-resampling perspective. *American Statistician*, 46:84–88.
Stephens, D. A. 1994. Bayesian retrospective multiple-changepoint identification. *Applied Statistics*, 43:159–178.

Stephens, D. A. and Smith, A. F. M. 1993. Bayesian inference in multipoint gene mapping. *Annals of Human Genetics*, 57:65–82.

Stephens, M. 2000. Bayesian analysis of mixture models with an unknown number of components—an alternative to reversible jump methods. *Annals of Statistics*, 28:40–74.

Tanner, M. and Wong, W. 1987. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82:528–550.

Tierney, L. 1994. Markov chains for exploring posterior distributions (with discussion). *Annals of Statistics*, 22:1701–1786.

Titterington, D. and Cox, D. (eds) 2001. *Biometrika: One Hundred Years*. Oxford University Press, Oxford.

Wakefield, J., Smith, A., Racine-Poon, A., and Gelfand, A. 1994. Bayesian analysis of linear and non-linear population models using the Gibbs sampler. *Applied Statistics*, 43:201–222.

Wang, C. S., Rutledge, J. J., and Gianola, D. 1993. Marginal inferences about variance-components in a mixed linear model using Gibbs sampling. *Genetics, Selection, Evolution*, 25:41–62.

Wang, C. S., Rutledge, J. J., and Gianola, D. 1994. Bayesian analysis of mixed limear models via Gibbs sampling with an application to litter size in Iberian pigs. *Genetics, Selection, Evolution*, 26:91–115.

Wei, G. and Tanner, M. 1990. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithm. *Journal of the American Statistical Association*, 85:699–704.

Zeger, S. and Karim, R. 1991. Generalized linear models with random effects; a Gibbs sampling approach. *Journal of the American Statistical Association*, 86:79–86.

3

Reversible Jump MCMC

Yanan Fan and Scott A. Sisson

3.1 Introduction

The reversible jump Markov chain Monte Carlo (MCMC) sampler (Green, 1995) provides a general framework for Markov chain Monte Carlo simulation in which the dimension of the parameter space can vary between iterates of the Markov chain. The reversible jump sampler can be viewed as an extension of the Metropolis–Hastings algorithm onto more general state spaces.

To understand this in a Bayesian modeling context, suppose that for observed data $\mathbf{x}$ we have a countable collection of candidate models $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots\}$ indexed by a parameter $k \in \mathcal{K}$. The index k can be considered as an auxiliary model indicator variable, such that $\mathcal{M}_{k'}$ denotes the model where $k = k'$. Each model $\mathcal{M}_k$ has an n_k-dimensional vector of unknown parameters, $\boldsymbol{\theta}_k \in \mathbb{R}^{n_k}$, where n_k can take different values for different models $k \in \mathcal{K}$. The joint posterior distribution of $(k, \boldsymbol{\theta}_k)$ given observed data, $\mathbf{x}$, is obtained as the product of the likelihood, $L(\mathbf{x} \mid k, \boldsymbol{\theta}_k)$, and the joint prior, $p(k, \boldsymbol{\theta}_k) = p(\boldsymbol{\theta}_k \mid k)p(k)$, constructed from the prior distribution of $\boldsymbol{\theta}_k$ under model $\mathcal{M}_k$, and the prior for the model indicator k (i.e. the prior for model $\mathcal{M}_k$). Hence, the joint posterior is

$$\pi(k, \boldsymbol{\theta}_k \mid \mathbf{x}) = \frac{L(\mathbf{x} \mid k, \boldsymbol{\theta}_k)p(\boldsymbol{\theta}_k \mid k)p(k)}{\sum_{k' \in \mathcal{K}} \int_{\mathbb{R}^{n_{k'}}} L(\mathbf{x} \mid k', \boldsymbol{\theta}'_{k'})p(\boldsymbol{\theta}'_{k'} \mid k')p(k')d\boldsymbol{\theta}'_{k'}}. \tag{3.1}$$

The reversible jump algorithm uses the joint posterior distribution in Equation 3.1 as the target of an MCMC sampler over the state space $\boldsymbol{\Theta} = \bigcup_{k \in \mathcal{K}}(\{k\} \times \mathbb{R}^{n_k})$, where the states of the Markov chain are of the form $(k, \boldsymbol{\theta}_k)$, the dimension of which can vary over the state space. Accordingly, from the output of a *single* Markov chain sampler, the user is able to obtain a full probabilistic description of the posterior probabilities of each model having observed the data, $\mathbf{x}$, in addition to the posterior distributions of the individual models.

This chapter aims to provide an overview of the reversible jump sampler. We will outline the sampler's theoretical underpinnings, present the latest and most popular techniques for enhancing algorithm performance, and discuss the analysis of sampler output. Through the use of numerous worked examples it is hoped that the reader will gain a broad appreciation of the issues involved in multi-model simulation, and the confidence to implement reversible jump samplers in the course of their own studies.

3.1.1 From Metropolis–Hastings to Reversible Jump

The standard formulation of the Metropolis–Hastings algorithm (Hastings, 1970) relies on the construction of a time-reversible Markov chain via the *detailed balance* condition. This

condition means that moves from state $\boldsymbol{\theta}$ to $\boldsymbol{\theta}'$ are made as often as moves from $\boldsymbol{\theta}'$ to $\boldsymbol{\theta}$ with respect to the target density. This is a simple way to ensure that the equilibrium distribution of the chain is the desired target distribution. The extension of the Metropolis–Hastings algorithm to the setting where the dimension of the parameter vector varies is more challenging theoretically, but the resulting algorithm is surprisingly simple to follow.

For the construction of a Markov chain on a general state space $\boldsymbol{\Theta}$ with invariant or stationary distribution π, the detailed balance condition can be written as

$$\int_{(\boldsymbol{\theta},\boldsymbol{\theta}')\in\mathcal{A}\times\mathcal{B}} \pi(d\boldsymbol{\theta})P(\boldsymbol{\theta}, d\boldsymbol{\theta}') = \int_{(\boldsymbol{\theta},\boldsymbol{\theta}')\in\mathcal{A}\times\mathcal{B}} \pi(d\boldsymbol{\theta}')P(\boldsymbol{\theta}', d\boldsymbol{\theta}) \tag{3.2}$$

for all Borel sets $\mathcal{A} \times \mathcal{B} \subset \boldsymbol{\Theta}$, where P is a general Markov transition kernel (Green, 2001).

As with the standard Metropolis–Hastings algorithm, Markov chain transitions from a current state $\boldsymbol{\theta} = (k, \boldsymbol{\theta}_k') \in \mathcal{A}$ in model $\mathcal{M}_k$ are realized by first proposing a new state $\boldsymbol{\theta}' = (k', \boldsymbol{\theta}_{k'}) \in \mathcal{B}$ in model $\mathcal{M}_{k'}$ from a proposal distribution $q(\boldsymbol{\theta}, \boldsymbol{\theta}')$. The detailed balance condition (Equation 3.2) is enforced through the acceptance probability, where the move to the candidate state $\boldsymbol{\theta}'$ is accepted with probability $\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}')$. If rejected, the chain remains at the current state $\boldsymbol{\theta}$ in model $\mathcal{M}_k$. Under this mechanism (Green, 2001, 2003), Equation 3.2 becomes

$$\int_{(\boldsymbol{\theta},\boldsymbol{\theta}')\in\mathcal{A}\times\mathcal{B}} \pi(\boldsymbol{\theta} \mid \mathbf{x})q(\boldsymbol{\theta}, \boldsymbol{\theta}')\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}')d\boldsymbol{\theta}\, d\boldsymbol{\theta}' = \int_{(\boldsymbol{\theta},\boldsymbol{\theta}')\in\mathcal{A}\times\mathcal{B}} \pi(\boldsymbol{\theta}' \mid \mathbf{x})q(\boldsymbol{\theta}', \boldsymbol{\theta})\alpha(\boldsymbol{\theta}', \boldsymbol{\theta})d\boldsymbol{\theta}\, d\boldsymbol{\theta}', \tag{3.3}$$

where the distributions $\pi(\boldsymbol{\theta} \mid \mathbf{x})$ and $\pi(\boldsymbol{\theta}' \mid \mathbf{x})$ are posterior distributions with respect to model $\mathcal{M}_k$ and $\mathcal{M}_{k'}$, respectively.

One way to enforce Equation 3.3 is by setting the acceptance probability as

$$\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}') = \min\left\{1, \frac{\pi(\boldsymbol{\theta} \mid \mathbf{x})q(\boldsymbol{\theta}, \boldsymbol{\theta}')}{\pi(\boldsymbol{\theta}' \mid \mathbf{x})q(\boldsymbol{\theta}', \boldsymbol{\theta})}\right\}, \tag{3.4}$$

where $\alpha(\boldsymbol{\theta}', \boldsymbol{\theta})$ is similarly defined. This resembles the usual Metropolis–Hastings acceptance ratio (Green, 1995; Tierney, 1998). It is straightforward to observe that this formulation includes the standard Metropolis–Hastings algorithm as a special case.

Accordingly, a reversible jump sampler with N iterations is commonly constructed as follows:

Step 1. Initialize k and $\boldsymbol{\theta}_k$ at iteration $t = 1$.

Step 2. For iteration $t \geq 1$ perform

- *Within-model move:* with a fixed model k, update the parameters $\boldsymbol{\theta}_k$ according to any MCMC updating scheme.
- *Between-models move:* simultaneously update model indicator k and the parameters $\boldsymbol{\theta}_k$ according to the general reversible proposal/acceptance mechanism (Equation 3.4).

Step 3. Increment iteration $t = t + 1$. If $t < N$, go to Step 2.

3.1.2 Application Areas

Statistical problems in which the number of unknown model parameters is itself unknown are extensive, and as such the reversible jump sampler has been implemented in analyses

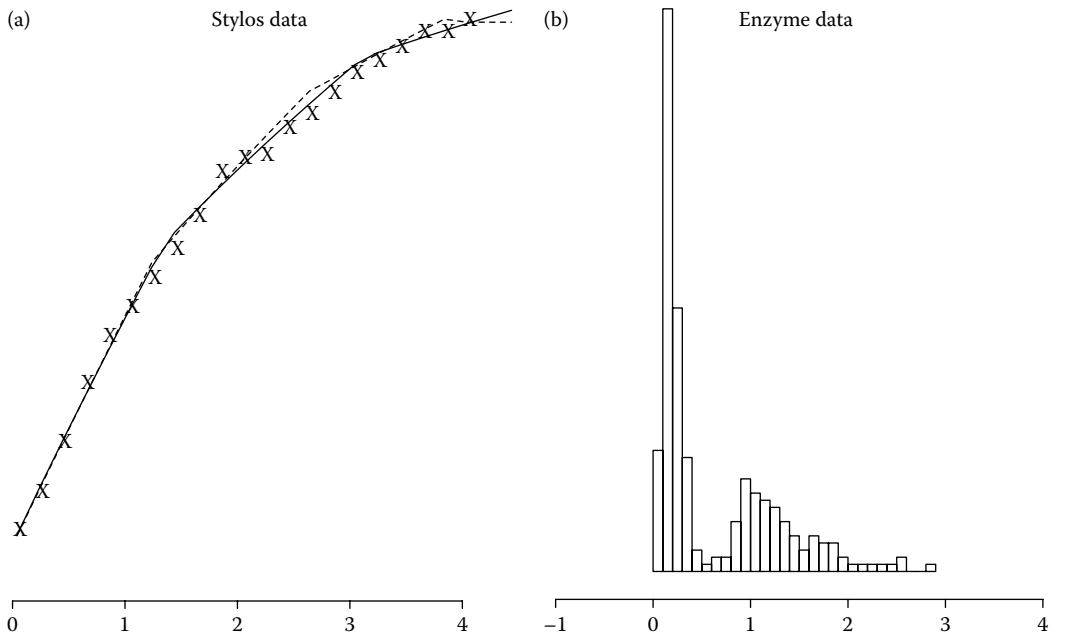

FIGURE 3.1
Examples of (a) change-point modeling and (b) mixture models. (a) With the Stylos tombs data set (crosses), a piecewise log-linear curve can be fitted between unknown change points. Illustrated are 2 (solid line) and 3 (dashed line) change points. (b) The histogram of the enzymatic activity data set suggests clear groupings of metabolizers, although the number of such groupings is not clear. (From Sisson, S. A. and Fan, Y. 2007. *Statistics and Computing*, 17:357–367. With permission.)

throughout a wide range of scientific disciplines over the last 15 years. Within the statistical literature, these predominantly concern Bayesian model determination problems (Sisson, 2005). Some of the commonly recurring models in this setting are described below.

Change-point models One of the original applications of the reversible jump sampler was in Bayesian change-point problems, where both the number and location of change points in a system is unknown *a priori*. For example, Green (1995) analyzed mining disaster count data using a Poisson process with the rate parameter described as a step function with an unknown number and location of steps. Fan and Brooks (2000) applied the reversible jump sampler to model the shape of prehistoric tombs, where the curvature of the dome changes an unknown number of times. Figure 3.1a shows the plot of depths and radii of one of the tombs from Crete in Greece. The data appear to be piecewise log-linear, with possibly two or three change points.

Finite mixture models Mixture models are commonly used where each data observation is generated according to some underlying categorical mechanism. This mechanism is typically unobserved, so there is uncertainty regarding which component of the resulting mixture distribution each data observation was generated from, in addition to uncertainty over the number of mixture components. A mixture model with k components for the observed data $\mathbf{x}$ takes the form

$$f(\mathbf{x} \mid \boldsymbol{\theta}_k) = \sum_{j=1}^{k} w_j f_j(\mathbf{x} \mid \boldsymbol{\phi}_j) \tag{3.5}$$

with $\boldsymbol{\theta}_k = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k)$, where w_j is the weight of the jth mixture component f_j, whose parameter vector is denoted by $\boldsymbol{\phi}_j$, and where $\sum_{j=1}^{k} w_j = 1$. The number of mixture components, k, is also unknown.

Figure 3.1b illustrates the distribution of enzymatic activity in the blood for 245 individuals. Richardson and Green (1997) analyzed these data using a mixture of normal densities to identify subgroups of slow or fast metabolizers. The multimodal nature of the data suggests the existence of such groups, but the number of distinct groupings is less clear. Tadesse et al. (2005) extend this normal mixture model for the purpose of clustering high-dimensional data.

Variable selection The problem of variable selection arises when modeling the relationship between a response variable, Y, and p potential explanatory variables $x_1, \ldots, x_p$. The multi-model setting emerges when attempting to identify the most relevant subsets of predictors, making it a natural candidate for the reversible jump sampler. For example, under a regression model with normal errors we have

$$Y = X_\gamma \beta_\gamma + \epsilon \quad \text{with } \epsilon \sim N(0, \sigma^2 I), \tag{3.6}$$

where $\gamma = (\gamma_1, \ldots, \gamma_p)$ is a binary vector indexing the subset of $x_1, \ldots, x_p$ to be included in the linear model, X_γ is the design matrix whose columns correspond to the indexed subset given by γ, and β_γ is the corresponding subset of regression coefficients. For examples and algorithms in this setting and beyond, see, for example, George and McCulloch (1993), Smith and Kohn (1996), and Nott and Leonte (2004).

Nonparametrics Within Bayesian nonparametrics, many authors have successfully explored the use of the reversible jump sampler as a method to automate the knot selection process when using a Pth-order spline model for curve fitting (Denison et al., 1998; DiMatteo et al., 2001). Here, a curve f is estimated by

$$f(x) = \alpha_0 + \sum_{j=1}^{P} \alpha_j x^j + \sum_{i=1}^{k} \eta_i (x - \kappa_i)_+^P, \quad x \in [a, b],$$

where $z_+ = \max(0, z)$ and κ_i, $i = 1, \ldots, k$, represent the locations of k knot points (Hastie and Tibshirani, 1990). Under this representation, fitting the curve consists of estimating the unknown number of knots k, the knot locations κ_i and the corresponding regression coefficients α_j and η_i, for $j = 0, \ldots, P$ and $i = 1, \ldots, k$.

Time series models In the modeling of temporally dependent data, $x_1, \ldots, x_T$, multiple models naturally arise under uncertainty over the degree of dependence. For example, under a kth-order autoregressive process

$$X_t = \sum_{\tau=1}^{k} a_\tau X_{t-\tau} + \epsilon_t, \quad \text{with } t = k+1, \ldots, T, \tag{3.7}$$

where $\epsilon_t \sim WN(0, \sigma^2)$, the order, k, of the autoregression is commonly unknown, in addition to the coefficients a_τ. Brooks et al. (2003c), Ehlers and Brooks (2003), and Vermaak et al. (2004) each detail descriptions on the use of reversible jump samplers for this class of problems.

The reversible jump algorithm has had a compelling influence in the statistical and mainstream scientific research literatures. In general, the large majority of application areas have tended to be computationally or biologically related (Sisson, 2005). Accordingly a large

number of developmental and application studies can be found in the signal processing literature and the related fields of computer vision and image analysis. Epidemiological and medical studies also feature strongly.

This chapter is structured as follows: In Section 3.2 we present a detailed description of how to implement the reversible jump sampler and review methods to improve sampler performance. Section 3.3 examines post-simulation analysis, including label switching problems when identifiability is an issue, and convergence assessment. In Section 3.4 we review related sampling methods in the statistical literature, and conclude with discussion on possible future research directions for the field. Other useful reviews of reversible jump MCMC can be found in Green (2003) and Sisson (2005).

3.2 Implementation

In practice, the construction of proposal moves between different models is achieved via the concept of "dimension matching." Most simply, under a general Bayesian model determination setting, suppose that we are currently in state $(k, \boldsymbol{\theta}_k)$ in model $\mathcal{M}_k$, and we wish to propose a move to a state $(k', \boldsymbol{\theta}'_{k'})$ in model $\mathcal{M}_{k'}$, which is of a higher dimension, so that $n_{k'} > n_k$. In order to "match dimensions" between the two model states, a random vector $\mathbf{u}$ of length $d_{k\to k'} = n_{k'} - n_k$ is generated from a known density $q_{d_{k\to k'}}(\mathbf{u})$. The current state $\boldsymbol{\theta}_k$ and the random vector $\mathbf{u}$ are then mapped to the new state $\boldsymbol{\theta}'_{k'} = g_{k\to k'}(\boldsymbol{\theta}_k, \mathbf{u})$ through a one-to-one mapping function $g_{k\to k'} : \mathbb{R}^{n_k} \times \mathbb{R}^{d_k} \to \mathbb{R}^{n_{k'}}$. The acceptance probability of this proposal, combined with the joint posterior expression of Equation 3.1, becomes

$$\alpha[(k, \boldsymbol{\theta}_k), (k', \boldsymbol{\theta}'_{k'})] = \min\left\{1, \frac{\pi(k', \boldsymbol{\theta}'_{k'} \mid \mathbf{x}) q(k' \to k)}{\pi(k, \boldsymbol{\theta}_k \mid \mathbf{x}) q(k \to k') q_{d_{k\to k'}}(\mathbf{u})} \left|\frac{\partial g_{k\to k'}(\boldsymbol{\theta}_k, \mathbf{u})}{\partial(\boldsymbol{\theta}_k, \mathbf{u})}\right|\right\}, \tag{3.8}$$

where $q(k \to k')$ denotes the probability of proposing a move from model $\mathcal{M}_k$ to model $\mathcal{M}_{k'}$, and the final term is the determinant of the Jacobian matrix, often referred to in the reversible jump literature simply as the Jacobian. This term arises through the change of variables via the function $g_{k\to k'}$, which is required when used with respect to the integral equation (Equation 3.3). Note that the normalization constant in Equation 3.1 is not needed to evaluate the above ratio. The reverse move proposal, from model $\mathcal{M}_{k'}$ to $\mathcal{M}_k$, is made deterministically in this setting, and is accepted with probability

$$\alpha\left[(k', \boldsymbol{\theta}'_{k'}), (k, \boldsymbol{\theta}_k)\right] = \alpha\left[(k, \boldsymbol{\theta}_k), (k', \boldsymbol{\theta}'_{k'})\right]^{-1}.$$

More generally, we can relax the condition on the length of the vector $\mathbf{u}$ by allowing $d_{k\to k'} \geq n_{k'} - n_k$. In this case, nondeterministic reverse moves can be made by generating a $d_{k'\to k}$-dimensional random vector $\mathbf{u}' \sim q_{d_{k'\to k}}(\mathbf{u}')$, such that the dimension matching condition, $n_k + d_{k\to k'} = n_{k'} + d_{k'\to k}$, is satisfied. Then a reverse mapping is given by $\boldsymbol{\theta}_k = g_{k'\to k}(\boldsymbol{\theta}'_{k'}, \mathbf{u}')$, such that $\boldsymbol{\theta}_k = g_{k'\to k}(g_{k\to k'}(\boldsymbol{\theta}_k, \mathbf{u}), \mathbf{u}')$ and $\boldsymbol{\theta}'_{k'} = g_{k\to k'}(g_{k'\to k}(\boldsymbol{\theta}'_{k'}, \mathbf{u}'), \mathbf{u})$. The acceptance probability corresponding to Equation 3.8 then becomes

$$\alpha\left[(k, \boldsymbol{\theta}_k), (k', \boldsymbol{\theta}'_{k'})\right] = \min\left\{1, \frac{\pi(k', \boldsymbol{\theta}'_{k'} \mid \mathbf{x}) q(k' \to k) q_{d_{k'\to k}}(\mathbf{u}')}{\pi(k, \boldsymbol{\theta}_k \mid \mathbf{x}) q(k \to k') q_{d_{k\to k'}}(\mathbf{u})} \left|\frac{\partial g_{k\to k'}(\boldsymbol{\theta}_k, \mathbf{u})}{\partial(\boldsymbol{\theta}_k, \mathbf{u})}\right|\right\}. \tag{3.9}$$

Example: Dimension Matching

Consider the illustrative example given in Green (1995) and Brooks (1998). Suppose that model $\mathcal{M}_1$ has states $(k = 1, \boldsymbol{\theta}_1 \in \mathbb{R}^1)$ and model $\mathcal{M}_2$ has states $(k = 2, \boldsymbol{\theta}_2 \in \mathbb{R}^2)$. Let $(1, \theta^*)$ denote the current state in $\mathcal{M}_1$ and $(2, (\theta^{(1)}, \theta^{(2)}))$ denote the proposed state in $\mathcal{M}_2$. Under dimension matching, we might generate a random scalar u, and let $\theta^{(1)} = \theta^* + u$ and $\theta^{(2)} = \theta^* - u$, with the reverse move given deterministically by $\theta^* = \frac{1}{2}\left(\theta^{(1)} + \theta^{(2)}\right)$.

Example: Moment Matching in a Finite Mixture of Univariate Normals

Under the finite mixture of univariate normals model, the observed data, $\mathbf{x}$, has density given by Equation 3.5, where the jth mixture component $f_j(\mathbf{x} \mid \boldsymbol{\phi}_j) = \phi(\mathbf{x} \mid \mu_j, \sigma_j)$ is the $N(\mu_j, \sigma_j)$ density. For between-model moves, Richardson and Green (1997) implement a split (one component into two) and merge (two components into one) strategy which satisfies the dimension matching requirement. (See Dellaportas and Papageorgiou (2006) for an alternative approach.)

When two normal components j_1 and j_2 are merged into one, j^*, Richardson and Green (1997) propose a deterministic mapping which maintains the zeroth, first, and second moments:

$$\begin{aligned} w_{j^*} &= w_{j_1} + w_{j_2}. \\ w_{j^*}\mu_{j^*} &= w_{j_1}\mu_{j_1} + w_{j_2}\mu_{j_2}. \\ w_{j^*}\left(\mu_{j^*}^2 + \sigma_{j^*}^2\right) &= w_{j_1}\left(\mu_{j_1}^2 + \sigma_{j_1}^2\right) + w_{j_2}\left(\mu_{j_2}^2 + \sigma_{j_2}^2\right). \end{aligned} \tag{3.10}$$

The split move is proposed as

$$\begin{aligned} w_{j_1} &= w_{j^*} * u_1, \quad w_{j_2} = w_{j^*} * (1 - u_1) \\ \mu_{j_1} &= \mu_{j^*} - u_2\sigma_{j^*}\sqrt{\frac{w_{j_2}}{w_{j_1}}} \\ \mu_{j_2} &= \mu_{j^*} + u_2\sigma_{j^*}\sqrt{\frac{w_{j_1}}{w_{j_2}}} \\ \sigma_{j_1}^2 &= u_3\left(1 - u_2^2\right)\sigma_{j^*}^2\frac{w_{j^*}}{w_{j_1}} \\ \sigma_{j_2}^2 &= (1 - u_3)\left(1 - u_2^2\right)\sigma_{j^*}^2\frac{w_{j^*}}{w_{j_2}}, \end{aligned} \tag{3.11}$$

where the random scalars $u_1, u_2 \sim \text{Beta}(2, 2)$ and $u_3 \sim \text{Beta}(1, 1)$. In this manner, dimension matching is satisfied, and the acceptance probability for the split move is calculated according to Equation 3.8, with the acceptance probability of the reverse merge move given by the reciprocal of this value.

3.2.1 Mapping Functions and Proposal Distributions

While the ideas behind dimension matching are conceptually simple, their implementation is complicated by the arbitrariness of the mapping function $g_{k\to k'}$ and the proposal distributions, $q_{d_{k\to k'}}(\mathbf{u})$, for the random vectors $\mathbf{u}$. Since mapping functions effectively express

functional relationships between the parameters of different models, good mapping functions will clearly improve sampler performance in terms of between-model acceptance rates and chain mixing. The difficulty is that even in the simpler setting of nested models, good relationships can be hard to define, and in more general settings, parameter vectors between models may not be obviously comparable.

The only additional degree of freedom to improve between-model proposals is by choosing the form and parameters of the proposal distribution $q_{d_{k\to k'}}(\mathbf{u})$. However, there are no obvious criteria to guide this choice. Contrast this to within-model, random-walk Metropolis–Hastings moves on a continuous target density, whereby proposed moves close to the current state can have an arbitrarily large acceptance probability, and proposed moves far from the current state have low acceptance probabilities. This concept of "local" moves may be partially translated on to model space ($k \in \mathcal{K}$): proposals from $\boldsymbol{\theta}_k$ in model $\mathcal{M}_k$ to $\boldsymbol{\theta}'_{k'}$ in model $\mathcal{M}_{k'}$ will tend to have larger acceptance probabilities if their likelihood values are similar, that is, $L(\mathbf{x} \mid k, \boldsymbol{\theta}_k) \approx L(\mathbf{x} \mid k', \boldsymbol{\theta}'_{k'})$. For example, in the analysis of Bayesian mixture models, Richardson and Green (1997) propose "birth/death" and "split/merge" mappings of mixture components for the between-model move, while keeping other components unchanged. Hence, the proposed moves necessarily will have similar likelihood values to the current state. However, in general the notion of "local" move proposals does not easily extend to the parameter vectors of different models, unless considering simplified settings (e.g. nested models). In the general case, good mixing properties are achieved by the alignment of regions of high posterior probability between models.

Notwithstanding these difficulties, reversible jump MCMC is often associated with poor sampler performance. However, failure to realize acceptable sampler performance should only be considered a result of poorly constructed between-model mappings or inappropriate proposal distributions. It should even be anticipated that implementing a multi-model sampler may result in improved chain mixing, even when the inferential target distribution is a single model. In this case, sampling from a single model posterior with an "overly sophisticated" machinery is loosely analogous to the extra performance gained with augmented state space sampling methods. For example, in the case of a finite mixture of normal distributions, Richardson and Green (1997) report markedly superior sampler mixing when conditioning on there being exactly three mixture components, in comparison with the output generated by a fixed-dimension sampler. George et al. (1999) similarly obtain improved chain performance in a single model, by performing "birth-then-death" moves simultaneously so that the dimension of the model remains constant. Green (2003) presents a short study on which inferential circumstances determine whether the adoption of a multi-model sampler may be beneficial in this manner. Conversely, Han and Carlin (2001) provide an argument to suggest that multi-model sampling may have a detrimental effect on efficiency.

3.2.2 Marginalization and Augmentation

Depending on the aim or the complexity of a multi-model analysis, it may be that use of reversible jump MCMC would be somewhat heavy-handed, when reduced- or fixed-dimensional samplers may be substituted. In some Bayesian model selection settings, between-model moves can be greatly simplified or even avoided if one is prepared to make certain prior assumptions, such as conjugacy or objective prior specifications. In such cases, it may be possible to analytically integrate out some or all of the parameters $\boldsymbol{\theta}_k$ in the posterior distribution (Equation 3.1), reducing the sampler either to fixed dimensions, for example on model space $k \in \mathcal{K}$ only, or to a lower-dimensional set of model and parameter

space (Berger and Pericchi, 2001; DiMatteo et al., 2001; George and McCulloch, 1993; Tadesse et al., 2005). In lower dimensions, the reversible jump sampler is often easier to implement, as the problems associated with mapping function specification are conceptually simpler to resolve.

Example: Marginalization in Variable Selection

In Bayesian variable selection for normal linear models (Equation 3.6), the vector $\gamma = (\gamma_1, \ldots, \gamma_p)$ is treated as an auxiliary (model indicator) variable, where

$$\gamma_i = \begin{cases} 1, & \text{if predictor } x_i \text{ is included in the regression,} \\ 0, & \text{otherwise.} \end{cases}$$

Under certain prior specifications for the regression coefficients β and error variance σ^2, the β coefficients can be analytically integrated out of the posterior. A Gibbs sampler directly on model space is then available for γ (George and McCulloch, 1993; Nott and Green, 2004; Smith and Kohn, 1996).

Example: Marginalization in Finite Mixture of Multivariate Normal Models

Within the context of clustering, the parameters of the normal components are usually not of interest. Tadesse et al. (2005) demonstrate that by choosing appropriate prior distributions, the parameters of the normal components can be analytically integrated out of the posterior. The reversible jump sampler may then run on a much reduced parameter space, which is simpler and more efficient.

In a general setting, Brooks et al. (2003c) proposed a class of models based on augmenting the state space of the target posterior with an auxiliary set of state-dependent variables, $\mathbf{v}_k$, so that the state space of $\pi(k, \boldsymbol{\theta}_k, \mathbf{v}_k \mid \mathbf{x}) = \pi(k, \boldsymbol{\theta}_k \mid \mathbf{x})\tau_k(\mathbf{v}_k)$ is of constant dimension for all models $\mathcal{M}_k \in \mathcal{M}$. By updating $\mathbf{v}_k$ via a (deliberately) slowly mixing Markov chain, a temporal memory is induced that persists in the $\mathbf{v}_k$ from state to state. In this manner, the motivation behind the auxiliary variables is to improve between-model proposals, in that some memory of previous model states is retained. Brooks et al. (2003c) demonstrate that this approach can significantly enhance mixing compared to an unassisted reversible jump sampler. Although the fixed dimensionality of $(k, \boldsymbol{\theta}_k, \mathbf{v}_k)$ is later relaxed, there is an obvious analogue with product space sampling frameworks (Carlin and Chib, 1995; Godsill, 2001); see Section 3.4.2.

An alternative augmented state space modification of standard MCMC is given by Liu et al. (2001). The dynamic weighting algorithm augments the original state space by a weighting factor, which permits the Markov chain to make large transitions not allowable by the standard transition rules, subject to the computation of the correct weighting factor. Inference is then made by using the weights to compute importance sampling estimates rather than simple Monte Carlo estimates. This method can be used within the reversible jump algorithm to facilitate cross-model jumps.

3.2.3 Centering and Order Methods

Brooks et al. (2003c) introduce a class of methods to achieve the automatic scaling of the proposal density, $q_{d_{k\to k'}}(\mathbf{u})$, based on "local" move proposal distributions, which are centered around the point of equal likelihood values under current and proposed models. Under this scheme, it is assumed that local mapping functions $g_{k\to k'}$ are known. For a proposed move from $(k, \boldsymbol{\theta}_k)$ in $\mathcal{M}_k$ to model $\mathcal{M}_{k'}$, the random vector "centering point"

$c_{k\to k'}(\boldsymbol{\theta}_k) = g_{k\to k'}(\boldsymbol{\theta}_k, \mathbf{u})$ is defined such that, for some particular choice of proposal vector $\mathbf{u}$, the current and proposed states are identical in terms of likelihood contribution, that is, $L(\mathbf{x} \mid k, \boldsymbol{\theta}_k) = L(\mathbf{x} \mid k', c_{k\to k'}(\boldsymbol{\theta}_k))$. For example, if $\mathcal{M}_k$ is an autoregressive model of order k (Equation 3.7) and $\mathcal{M}_{k'}$ is an autoregressive model of order $k' = k + 1$, and if $c_{k\to k'}(\boldsymbol{\theta}_k) = g_{k\to k'}(\boldsymbol{\theta}_k, u) = (\boldsymbol{\theta}_k, u)$ (e.g. a local "birth" proposal), then we have $u = 0$ and $c_{k\to k'} = (\boldsymbol{\theta}_k, 0)$, as $L(\mathbf{x} \mid k, \boldsymbol{\theta}_k) = L(\mathbf{x} \mid k', (\boldsymbol{\theta}_k, 0))$.

Given the centering constraint on $\mathbf{u}$, if the scaling parameter in the proposal $q_{d_{k\to k'}}(\mathbf{u})$ is a scalar, then the zeroth-order method (Brooks et al., 2003c) proposes to choose this scaling parameter such that the acceptance probability $\alpha[(k, \boldsymbol{\theta}_k), (k', c_{k\to k'}(\boldsymbol{\theta}_k))]$ of a move to the centering point $c_{k\to k'}(\boldsymbol{\theta}_k)$ in model $\mathcal{M}_{k'}$ is exactly one. The argument is then that move proposals close to $c_{k\to k'}(\boldsymbol{\theta}_k)$ will also have a large acceptance probability.

For proposal distributions, $q_{d_{k\to k'}}(\mathbf{u})$, with additional degrees of freedom, a similar method based on a series of nth-order conditions (for $n \geq 1$) requires that, for the proposed move, the nth derivative (with respect to $\mathbf{u}$) of the acceptance probability equals the zero vector at the centering point $c_{k\to k'}(\boldsymbol{\theta}_k)$:

$$\nabla^n \alpha[(k, \boldsymbol{\theta}_k), (k', c_{k\to k'}(\boldsymbol{\theta}_k))] = \mathbf{0}. \tag{3.12}$$

That is, the m unknown parameters in the proposal distribution $q_{d_{k\to k'}}(\mathbf{u})$ are determined by solving the m simultaneous equations given by Equation 3.12 with $n = 1, \ldots, m$. The idea behind the nth-order method is that the concept of closeness to the centering point under the zeroth-order method is relaxed. By enforcing zero derivatives of $\alpha[(k, \boldsymbol{\theta}_k), (k', c_{k\to k'}(\boldsymbol{\theta}_k))]$, the acceptance probability will become flatter around $c_{k\to k'}(\boldsymbol{\theta}_k)$. Accordingly this allows proposals further away from the centering point to still be accepted with a reasonably high probability. This will ultimately induce improved chain mixing.

With these methods, proposal distribution parameters are adapted to the current state of the chain, $(k, \boldsymbol{\theta}_k)$, rather than relying on a constant proposal parameter vector for all state transitions. It can be shown that for a simple two-model case, the nth-order conditions are optimal in terms of the capacitance of the algorithm (Lawler and Sokal, 1988). See also Ehlers and Brooks (2003) for an extension to a more general setting, and Ntzoufras et al. (2003) for a centering method in the context of linear models.

One caveat with the centering schemes is that they require specification of the between-model mapping function $g_{k\to k'}$, although these methods compensate for poor choices of mapping functions by selecting the best set of parameters for the given mapping. Recently, Ehlers and Brooks (2008) suggest the posterior conditional distribution $\pi(k', \mathbf{u} \mid \boldsymbol{\theta}_k)$ as the proposal for the random vector $\mathbf{u}$, side-stepping the need to construct a mapping function. In this case, the full conditionals either must be known or need to be approximated.

Example: The Zeroth-Order Method for an Autoregressive Model

Brooks et al. (2003c) consider the AR model with unknown order k (Equation 3.7), assuming Gaussian noise $\epsilon_t \sim N(0, \sigma_\epsilon^2)$ and a uniform prior on k, where $k = 1, 2, \ldots, k_{\max}$. Within each model $\mathcal{M}_k$, independent $N(0, \sigma_a^2)$ priors are adopted for the AR coefficients a_τ, $\tau = 1, \ldots, k$, with an inverse gamma prior for σ_ϵ^2. Suppose moves are made from model $\mathcal{M}_k$ to model $\mathcal{M}_{k'}$ such that $k' = k + 1$. The move from $\boldsymbol{\theta}_k$ to $\boldsymbol{\theta}'_{k'}$ is achieved by generating a random scalar $u \sim q(u) = N(0, 1)$, and defining the mapping function as $\boldsymbol{\theta}'_{k'} = g_{k\to k'}(\boldsymbol{\theta}_k, u) = (\boldsymbol{\theta}_k, \sigma u)$. The centering point $c_{k\to k'}(\boldsymbol{\theta}_k)$ then occurs at the point $u = 0$, or $\boldsymbol{\theta}'_{k'} = (\boldsymbol{\theta}_k, 0)$.

Under the mapping $g_{k\to k'}$, the Jacobian is σ, and the acceptance probability (Equation 3.8) for the move from $(k, \boldsymbol{\theta}_k)$ to $(k', c_{k\to k'}(\boldsymbol{\theta}_k))$ is given by $\alpha[(k, \boldsymbol{\theta}_k), (k', (\boldsymbol{\theta}_k, 0))] = \min(1, A)$ where

$$A = \frac{\pi(k', (\boldsymbol{\theta}_k, 0) \mid \mathbf{x}) q(k' \to k)\sigma}{\pi(k, \boldsymbol{\theta}_k \mid \mathbf{x}) q(k \to k') q(0)} = \frac{\left(2\pi\sigma_a^2\right)^{-1/2} q(k' \to k)\sigma}{q(k \to k')(2\pi)^{-1/2}}.$$

Note that since the likelihoods are equal at the centering point, and the priors common to both models cancel in the posterior ratio, A is only a function of the prior density for the parameter a_{k+1} evaluated at 0, the proposal distributions and the Jacobian. Hence, we solve $A = 1$ to obtain

$$\sigma^2 = \sigma_a^2 \left(\frac{q(k \to k')}{q(k' \to k)}\right)^2.$$

Thus in this case, the proposal variance is not dependent on the model parameter $(\boldsymbol{\theta}_k)$ or data $(\mathbf{x})$. It depends only on the prior variance, σ_a, and the model states, k, k'.

Example: The Second-Order Method for Moment Matching

Consider the moment matching in a finite mixture of univariate normals example of Section 3.2. The mapping functions $g_{k'\to k}$ and $g_{k\to k'}$ are respectively given by Equations 3.10 and 3.11, with the random numbers u_1, u_2, u_3 drawn from independent beta distributions with unknown parameter values, so that $q_{p_i,q_i}(u_i)$: $u_i \sim \text{Beta}(p_i, q_i)$, $i = 1, 2, 3$.

Consider the split move, Equation 3.11. To apply the second-order method of Brooks et al. (2003c), we first locate a centering point, $c_{k\to k'}(\boldsymbol{\theta}_k)$, achieved by setting $u_1 = 1$, $u_2 = 0$, and $u_3 \equiv u_1 = 1$ by inspection. Hence, at the centering point, the two new (split) components j_1 and j_2 will have the same location and scale as the j^* component, with new weights $w_{j_1} = w_{j^*}$ and $w_{j_2} = 0$ and all observations allocated to component j_1. Accordingly this will produce identical likelihood contributions. Note that to obtain equal variances for the split proposal, substitute the expressions for w_{j_1} and w_{j_2} into those for $\sigma_{j_1}^2 = \sigma_{j_2}^2$.

Following Richardson and Green (1997), the acceptance probability of the split move evaluated at the centering point is then proportional (with respect to $\mathbf{u}$) to

$$\begin{aligned}
&\log A[(k, \boldsymbol{\theta}_k), (k', c_{k\to k'}(\boldsymbol{\theta}_k))] \\
&\propto l_{j_1} \log(w_{j_1}) + l_{j_2} \log(w_{j_2}) - \frac{l_{j_1}}{2} \log\left(\sigma_{j_1}^2\right) - \frac{l_{j_2}}{2} \log\left(\sigma_{j_2}^2\right) - \frac{1}{2\sigma_{j_1}^2} \sum_{l=1}^{l_{j_1}} (y_l - \mu_{j_1})^2 \\
&- \frac{1}{2\sigma_{j_2}^2} \sum_{l=1}^{l_{j_2}} (y_l - \mu_{j_2})^2 + (\delta - 1 + l_{j_1}) \log(w_{j_1}) + (\delta - 1 + l_{j_2}) \log(w_{j_2}) \\
&- \left\{\frac{1}{2}\kappa[(\mu_{j_1} - \xi)^2 + (\mu_{j_2} - \xi)^2]\right\} - (\alpha + 1) \log\left(\sigma_{j_1}^2 \sigma_{j_2}^2\right) - \beta\left(\sigma_{j_1}^{-2} + \sigma_{j_2}^{-2}\right) \\
&- \log[q_{p_1,q_1}(u_1)] - \log[q_{p_2,q_2}(u_2)] - \log[q_{p_3,q_3}(u_3)] + \log(|\mu_{j_1} - \mu_{j_2}|) \\
&+ \log\left(\sigma_{j_1}^2\right) + \log\left(\sigma_{j_2}^2\right) - \log(u_2) - \log\left(1 - u_2^2\right) - \log(u_3) - \log(1 - u_3), \qquad (3.13)
\end{aligned}$$

where l_{j_1} and l_{j_2} respectively denote the number of observations allocated to components j_1 and j_2, and where $\delta, \alpha, \beta, \xi$ and κ are hyperparameters as defined by Richardson and Green (1997).

Thus, for example, to obtain the proposal parameter values p_1 and q_1 for u_1, we solve the first- and second-order derivatives of the acceptance probability (Equation 3.13) with respect to u_1. This yields

$$\frac{\partial \log \alpha\left[(k,\boldsymbol{\theta}_k),(k',c_{k\to k'}(\boldsymbol{\theta}_k))\right]}{\partial u_1} = \frac{\delta + 2l_{j_1} - p_1}{u_1} + \frac{q_1 - \delta - 2l_{j_2}}{(1-u_1)}$$
$$\frac{\partial^2 \log \alpha\left[(k,\boldsymbol{\theta}_k),(k',c_{k\to k'}(\boldsymbol{\theta}_k))\right]}{\partial u_1^2} = -\frac{\delta + 2l_{j_1} - p_1}{u_1^2} + \frac{q_1 - \delta - 2l_{j_2}}{(1-u_1)^2}.$$

Equating these to zero and solving for p_1 and q_1 at the centering points (with $l_{j_1} = l_{j^*}$ and $l_{j_2} = 0$) gives $p_1 = \delta + 2l_{j^*}$ and $q_1 = \delta$. Thus the parameter p_1 depends on the number of observations allocated to the component being split. Similar calculations to the above give solutions for p_2, q_2, p_3, and q_3.

3.2.4 Multi-Step Proposals

Green and Mira (2001) introduce a procedure for learning from rejected between-model proposals based on an extension of the splitting rejection idea of Tierney and Mira (1999). After rejecting a between-model proposal, the procedure makes a second proposal, usually under a modified proposal mechanism, and potentially dependent on the value of the rejected proposal. In this manner, a limited form of adaptive behavior may be incorporated into the proposals. The procedure is implemented via a modified Metropolis–Hastings acceptance probability, and may be extended to more than one sequential rejection (Trias et al., 2009). Delayed-rejection schemes can reduce the asymptotic variance of ergodic averages by reducing the probability of the chain remaining in the same state (Peskun, 1973; Tierney, 1998), however there is an obvious tradeoff with the extra move construction and computation required.

For clarity of exposition, in the remainder of this section we denote the current state of the Markov chain in model $\mathcal{M}_k$ by $\mathbf{x} = (k, \boldsymbol{\theta}_k)$, and the first and second stage proposed states in model $\mathcal{M}_{k'}$ by $\mathbf{y}$ and $\mathbf{z}$. Let $\mathbf{y} = g^{(1)}_{k\to k'}(\mathbf{x}, \mathbf{u}_1)$ and $\mathbf{z} = g^{(2)}_{k\to k'}(\mathbf{x}, \mathbf{u}_1, \mathbf{u}_2)$ be the mappings of the current state and random vectors $\mathbf{u}_1 \sim q^{(1)}_{d_{k\to k'}}(\mathbf{u}_1)$ and $\mathbf{u}_2 \sim q^{(2)}_{d_{k\to k'}}(\mathbf{u}_2)$ into the proposed new states. For simplicity, we again consider the framework where the dimension of model $\mathcal{M}_k$ is smaller than that of model $\mathcal{M}_{k'}$ (i.e. $n_{k'} > n_k$) and where the reverse move proposals are deterministic. The proposal from $\mathbf{x}$ to $\mathbf{y}$ is accepted with the usual acceptance probability

$$\alpha_1(\mathbf{x},\mathbf{y}) = \min\left\{1, \frac{\pi(\mathbf{y})q(k'\to k)}{\pi(\mathbf{x})q(k\to k')q^{(1)}_{d_{k\to k'}}(\mathbf{u}_1)}\left|\frac{\partial g^{(1)}_{k\to k'}(\mathbf{x},\mathbf{u}_1)}{\partial(\mathbf{x},\mathbf{u}_1)}\right|\right\}.$$

If $\mathbf{y}$ is rejected, detailed balance for the move from $\mathbf{x}$ to $\mathbf{z}$ is preserved with the acceptance probability

$$\alpha_2(\mathbf{x},\mathbf{z}) = \min\left\{1, \frac{\pi(\mathbf{z})q(k'\to k)[1-\alpha_1(\mathbf{y}^*,\mathbf{z})^{-1}]}{\pi(\mathbf{x})q(k\to k')q^{(1)}_{d_{k\to k'}}(\mathbf{u}_1)q^{(2)}_{d_{k\to k'}}(\mathbf{u}_2)[1-\alpha_1(\mathbf{x},\mathbf{y})]}\left|\frac{\partial g^{(2)}_{k\to k'}(\mathbf{x},\mathbf{u}_1,\mathbf{u}_2)}{\partial(\mathbf{x},\mathbf{u}_1,\mathbf{u}_2)}\right|\right\},$$

where $\mathbf{y}^* = g^{(1)}_{k\to k'}(\mathbf{z},\mathbf{u}_1)$. Note that the second stage proposal $\mathbf{z} = g^{(2)}_{k\to k'}(\mathbf{x},\mathbf{u}_1,\mathbf{u}_2)$ is permitted to depend on the rejected first stage proposal $\mathbf{y}$ (a function of $\mathbf{x}$ and $\mathbf{u}_1$).

In a similar vein, Al-Awadhi et al. (2004) also acknowledge that an initial between-model proposal $\mathbf{x}' = g_{k\to k'}(\mathbf{x}, \mathbf{u})$ may be poor, and seek to adjust the state $\mathbf{x}'$ to a region of higher posterior probability before taking the decision to accept or reject the proposal. Specifically, Al-Awadhi et al. (2004) propose to initially evaluate the proposed move to $\mathbf{x}'$ in model $\mathcal{M}_{k'}$ through a density $\pi^*(\mathbf{x}')$ rather than the usual $\pi(\mathbf{x}')$. The authors suggest taking π^* to be some tempered distribution $\pi^* = \pi^\gamma$, $\gamma > 1$, such that the modes of π^* and π are aligned.

The algorithm then implements $\kappa \geq 1$ fixed-dimension MCMC updates, generating states $\mathbf{x}' \to \mathbf{x}^1 \to \cdots \to \mathbf{x}^\kappa = \mathbf{x}^*$, with each step satisfying detailed balance with respect to π^*. This provides an opportunity for $\mathbf{x}^*$ to move closer to the mode of π^* (and therefore π) than $\mathbf{x}'$. The move from $\mathbf{x}$ in model $\mathcal{M}_k$ to the final state $\mathbf{x}^*$ in model $\mathcal{M}_{k'}$ (with density $\pi(\mathbf{x}^*)$) is finally accepted with probability

$$\alpha(\mathbf{x}, \mathbf{x}^*) = \min\left\{1, \frac{\pi(\mathbf{x}^*)\pi^*(\mathbf{x}')q(k' \to k)}{\pi(\mathbf{x})\pi^*(\mathbf{x}^*)q(k \to k')q_{d_{k\to k'}}(\mathbf{u})}\left|\frac{\partial g_{k\to k'}(\mathbf{x}, \mathbf{u})}{\partial(\mathbf{x}, \mathbf{u})}\right|\right\}.$$

The implied reverse move from model $\mathcal{M}_{k'}$ to model model $\mathcal{M}_k$ is conducted by taking the κ moves with respect to π^* first, followed by the dimension-changing move.

Various extensions can easily be incorporated into this framework, such as using a sequence of π^* distributions, resulting in a slightly modified acceptance probability expression. For instance, the standard simulated annealing framework, Kirkpatrick (1984), provides an example of a sequence of distributions which encourage moves toward posterior mode. Clearly the choice of the distribution π^* can be crucial to the success of this strategy. As with all multi-step proposals, increased computational overheads are traded for potentially enhanced between-model mixing.

3.2.5 Generic Samplers

The problem of efficiently constructing between-model mapping templates, $g_{k\to k'}$, with associated random vector proposal densities, $q_{d_{k\to k'}}$, may be approached from an alternative perspective. Rather than relying on a user-specified mapping, one strategy would be to move toward a more generic proposal mechanism altogether. A clear benefit of generic between-model moves is that they may be equally be implemented for nonnested models. While the ideal of "black-box" between-model proposals is attractive, they currently remain on the research horizon. However, a number of automatic reversible jump MCMC samplers have been proposed.

Green (2003) proposed a reversible jump analogy of the random-walk Metropolis sampler of Roberts (2003). Suppose that estimates of the first- and second-order moments of $\boldsymbol{\theta}_k$ are available, for each of a small number of models, $k \in \mathcal{K}$, denoted by $\boldsymbol{\mu}_k$ and $\mathbf{B}_k\mathbf{B}_k^\top$ respectively, where $\mathbf{B}_k$ is an $n_k \times n_k$ matrix. In proposing a move from $(k, \boldsymbol{\theta}_k)$ to model $\mathcal{M}_{k'}$, a new parameter vector is proposed by

$$\boldsymbol{\theta}'_{k'} = \begin{cases} \boldsymbol{\mu}_{k'} + \mathbf{B}_{k'}\left[\mathbf{R}\mathbf{B}_k^{-1}(\boldsymbol{\theta}_k - \boldsymbol{\mu}_k)\right]_1^{n_{k'}}, & \text{if } n_{k'} < n_k, \\ \boldsymbol{\mu}_{k'} + \mathbf{B}_{k'}\mathbf{R}\mathbf{B}_k^{-1}(\boldsymbol{\theta}_k - \boldsymbol{\mu}_k), & \text{if } n_{k'} = n_k, \\ \boldsymbol{\mu}_{k'} + \mathbf{B}_{k'}\mathbf{R}\begin{pmatrix}\mathbf{B}_k^{-1}(\boldsymbol{\theta}_k - \boldsymbol{\mu}_k) \\ \mathbf{u}\end{pmatrix}, & \text{if } n_{k'} > n_k, \end{cases}$$

where $[\cdot]_1^m$ denotes the first m components of a vector, $\mathbf{R}$ is a orthogonal matrix of order $\max\{n_k, n_{k'}\}$, and $\mathbf{u} \sim q_{n_{k'}-n_k}(\mathbf{u})$ is an $(n_{k'} - n_k)$-dimensional random vector (only utilized

if $n_{k'} > n_k$, or when calculating the acceptance probability of the reverse move from model $\mathcal{M}_{k'}$ to model $\mathcal{M}_k$ if $n_{k'} < n_k$). If $n_{k'} \leq n_k$, then the proposal $\boldsymbol{\theta}'_{k'}$ is deterministic and the Jacobian is trivially calculated. Hence, the acceptance probability is given by

$$\alpha\left[(k, \boldsymbol{\theta}_k), (k', \boldsymbol{\theta}'_{k'})\right] = \frac{\pi(k', \boldsymbol{\theta}'_{k'} \mid \mathbf{x})}{\pi(k, \boldsymbol{\theta}_k \mid \mathbf{x})} \frac{q(k' \to k)}{q(k \to k')} \frac{|\mathbf{B}_{k'}|}{|\mathbf{B}_k|} \times \begin{cases} q_{n_{k'}-n_k}(\mathbf{u}), & \text{for } n_{k'} < n_k, \\ 1, & \text{for } n_{k'} = n_k, \\ \dfrac{1}{q_{n_{k'}-n_k}}(\mathbf{u}), & \text{for } n_{k'} > n_k. \end{cases}$$

Accordingly, if the model-specific densities $\pi(k, \boldsymbol{\theta}_k \mid \mathbf{x})$ are unimodal with first- and second-order moments given by $\boldsymbol{\mu}_k$ and $\mathbf{B}_k \mathbf{B}_k^\top$, then high between-model acceptance probabilities may be achieved. (Unitary acceptance probabilities are available if the $\pi(k, \boldsymbol{\theta}_k \mid \mathbf{x})$ are exactly Gaussian.) Green (2003), Godsill (2003), and Hastie (2004) discuss a number of modifications to this general framework, including improving efficiency and relaxing the requirement of unimodal densities $\pi(k, \boldsymbol{\theta}_k \mid \mathbf{x})$ to realize high between-model acceptance rates. Naturally, the required knowledge of first- and second-order moments of each model density will restrict the applicability of these approaches to moderate numbers of candidate models if these require estimation (e.g. via pilot chains).

With a similar motivation to the above, Papathomas et al. (2009) put forward the multivariate normal as proposal distribution for $\boldsymbol{\theta}'_{k'}$ in the context of linear regression models, so that $\boldsymbol{\theta}'_{k'} \sim N(\boldsymbol{\mu}_{k'|\boldsymbol{\theta}_k}, \boldsymbol{\Sigma}_{k'|\boldsymbol{\theta}_k})$. The authors derive estimates for the mean $\boldsymbol{\mu}_{k'|\boldsymbol{\theta}_k}$ and covariance $\boldsymbol{\Sigma}_{k'|\boldsymbol{\theta}_k}$ such that the proposed values for $\boldsymbol{\theta}'_{k'}$ will on average produce conditional posterior values under model $\mathcal{M}_{k'}$ similar to those produced by the vector $\boldsymbol{\theta}_k$ under model $\mathcal{M}_k$. In particular, consider the normal linear model in Equation 3.6, rewriting the error covariance as V, assuming equality under the two models such that $V_k = V_{k'} = V$. The parameters of the proposal distribution for $\boldsymbol{\theta}'_{k'}$ are then given by

$$\boldsymbol{\mu}_{k'|\boldsymbol{\theta}_k} = (X_{\gamma'}^\top V^{-1} X_{\gamma'})^{-1} X_{\gamma'}^\top V^{-1} \left\{ Y + B^{-1} V^{-1/2} (X_\gamma \boldsymbol{\theta}_k - P_k Y) \right\},$$

$$\boldsymbol{\Sigma}_{k'|\boldsymbol{\theta}_k} = Q_{k',k'} - Q_{k',k'} Q_{k',k}^{-1} Q_{k,k} Q_{k,k'}^{-1} Q_{k',k'} + cI_{n_{k'}},$$

where γ and γ' are indicators corresponding to models $\mathcal{M}_k$ and $\mathcal{M}_{k'}$, $B = (V + X_{\gamma'} \boldsymbol{\Sigma}_{k'|\boldsymbol{\theta}_k} X_{\gamma'}^\top)^{-1/2}$, $P_k = X_\gamma (X_\gamma^\top V^{-1} X_\gamma)^{-1} X_\gamma^\top V^{-1}$, $Q_{k,k'} = (X_\gamma^\top V^{-1} X_{\gamma'})^{-1}$, I_n is the $n \times n$ identity matrix and $c > 0$. Intuitively, the mean of this proposal distribution may be interpreted as the maximum likelihood estimate of $\boldsymbol{\theta}'_{k'}$ for model $\mathcal{M}_{k'}$, plus a correction term based on the distance of the current chain state $\boldsymbol{\theta}_k$ to the mode of the posterior density in model $\mathcal{M}_k$. The mapping between $\boldsymbol{\theta}'_{k'}$ and $\boldsymbol{\theta}_k$ and the random number $\mathbf{u}$ is given by

$$\boldsymbol{\theta}'_{k'} = \boldsymbol{\mu}_{k'|\boldsymbol{\theta}_k} + \boldsymbol{\Sigma}_{k'|\boldsymbol{\theta}_k}^{1/2} \mathbf{u},$$

where $\mathbf{u} \sim N(0, I_{n_{k'}})$. Accordingly the Jacobian corresponding to Equation 3.9 is given by $\left|\boldsymbol{\Sigma}_{k'|\boldsymbol{\theta}_k}^{1/2}\right| \left|\boldsymbol{\Sigma}_{k|\boldsymbol{\theta}_{k'}}^{1/2}\right|$. Under this construction, the value $c > 0$ is treated as a tuning parameter for the calibration of the acceptance probability. Quite clearly, the parameters of the between-model proposal do not require *a priori* estimation, and they adapt to the current state of the chain. The authors note that in some instances, this method produces similar results in terms of efficiency to Green (2003). One caveat is that the calculations at each proposal stage involve several inversions of matrices which can be computationally costly when the dimension is large. In addition, the method is theoretically justified for normal linear

models, but can be applied to nonnormal models when transformation of data to normality is available, as demonstrated in Papathomas et al. (2009).

Fan et al. (2009) propose to construct between-model proposals based on estimating conditional marginal densities. Suppose that it is reasonable to assume some structural similarities between the parameters $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}'_{k'}$ of models $\mathcal{M}_k$ and $\mathcal{M}_{k'}$, respectively. Let c indicate the subset of the vectors $\boldsymbol{\theta}_k = (\boldsymbol{\theta}_k^c, \boldsymbol{\theta}_k^{-c})$ and $\boldsymbol{\theta}'_{k'} = (\boldsymbol{\theta}_{k'}^c, \boldsymbol{\theta}_{k'}^{-c})$ which can be kept constant between models, so that $\boldsymbol{\theta}_{k'}^c = \boldsymbol{\theta}_k^c$. The remaining r-dimensional vector $\boldsymbol{\theta}_{k'}^{-c}$ is then sampled from an estimate of the factorization of the conditional posterior of $\boldsymbol{\theta}_{k'}^{-c} = (\theta_{k'}^1, \ldots, \theta_{k'}^r)$ under model $\mathcal{M}_{k'}$:

$$\pi(\boldsymbol{\theta}_{k'}^{-c} \mid \boldsymbol{\theta}_{k'}^c, \mathbf{x}) \approx \hat{\pi}_1(\theta_{k'}^1 \mid \theta_{k'}^2, \ldots, \theta_{k'}^r, \boldsymbol{\theta}_{k'}^c, \mathbf{x}) \ldots \hat{\pi}_{r-1}(\theta_{k'}^{r-1} \mid \theta_{k'}^r, \boldsymbol{\theta}_{k'}^c, \mathbf{x}) \hat{\pi}_r(\theta_{k'}^r \mid \boldsymbol{\theta}_{k'}^c, \mathbf{x}).$$

The proposal $\boldsymbol{\theta}_{k'}^{-c}$ is drawn by first estimating $\hat{\pi}_r(\theta_{k'}^r \mid \boldsymbol{\theta}_{k'}^c, \mathbf{x})$ and sampling $\theta_{k'}^r$, and by then estimating $\hat{\pi}_{r-1}(\theta_{k'}^{r-1} \mid \theta_{k'}^r, \boldsymbol{\theta}_{k'}^c, \mathbf{x})$ and sampling $\theta_{k'}^{r-1}$, conditioning on the previously sampled point, $\theta_{k'}^r$, and so on. Fan et al. (2009) construct the conditional marginal densities by using partial derivatives of the joint density, $\pi(k', \boldsymbol{\theta}'_{k'} \mid \mathbf{x})$, to provide gradient information within a marginal density estimator. As the conditional marginal density estimators are constructed using a combination of samples from the prior distribution and gridded values, they can be computationally expensive to construct, particularly if high-dimensional moves are attempted, for example $\boldsymbol{\theta}_{k'}^{-c} = \boldsymbol{\theta}'_{k'}$. However, this approach can be efficient, and also adapts to the current state of the sampler.

3.3 Post Simulation

3.3.1 Label Switching

The so-called "label switching" problem occurs when the posterior distribution is invariant under permutations in the labeling of the parameters. This results in the parameters having identical marginal posterior distributions. For example, in the context of a finite mixture model (Equation 3.5), the parameters of each mixture component, $\boldsymbol{\phi}_j$, are unidentifiable under a symmetric prior. This causes problems in the interpretation of the MCMC output. While this problem is general, in that it is not restricted to the multi-model case, as many applications of the reversible jump sampler encounter this type of problem, we discuss some methods of overcoming this issue below.

The conceptually simplest method of circumventing nonidentifiability is to impose artificial constraints on the parameters. For example, if μ_j denotes the mean of the jth Gaussian mixture component, then one such constraint could be $\mu_1 < \cdots < \mu_k$ (Richardson and Green, 1997). However, the effectiveness of this approach is not always guaranteed (Jasra et al., 2005). One of the main problems with such constraints is that they are often artificial, being imposed for inferential convenience rather than as a result of genuine knowledge about the model. Furthermore, suitable constraints can be difficult or almost impossible to find (Frühwirth-Schnatter, 2001).

Alternative approaches to handling nonidentifiability involve the post-processing of MCMC output. Stephens (2000b) gives an inferential method based on the relabeling of components with respect to the permutation which minimizes the posterior expected loss. Celeux et al. (2000), Hurn et al. (2003), and Sisson and Hurn (2004) adopt a fully decision-theoretic approach, where for every posterior quantity of interest, an appropriate (possibly

multi-model) loss function is constructed and minimized. Each of these methods can be computationally expensive.

3.3.2 Convergence Assessment

Under the assumption that an acceptably efficient method of constructing a reversible jump sampler is available, one obvious pre-requisite to inference is that the Markov chain converges to its equilibrium state. Even in fixed dimension problems, theoretical convergence bounds are in general difficult or impossible to determine. In the absence of such theoretical results, convergence diagnostics based on empirical statistics computed from the sample path of multiple chains are often the only available tool. An obvious drawback of the empirical approach is that such diagnostics invariably fail to detect a lack of convergence when parts of the target distribution are missed entirely by all replicate chains. Accordingly, these are necessary rather than sufficient indicators of chain convergence; see Mengersen et al. (1999) and Cowles and Carlin (1996) for comparative reviews under fixed dimension MCMC.

The reversible jump sampler generates additional problems in the design of suitable empirical diagnostics, since most of these depend on the identification of suitable scalar statistics of the parameters' sample paths. However, in the multi-model case, these parameters may no longer retain the same interpretation. In addition, convergence is required not only within each of a potentially large number of models, but also across models with respect to posterior model probabilities.

One obvious approach would be the implementation of independent sub-chain assessments, both within models and for the model indicator $k \in \mathcal{K}$. With focus purely on model selection, Brooks et al. (2003b) propose various diagnostics based on the sample path of the model indicator, k, including nonparametric hypothesis tests such as the χ^2 and Kolmogorov–Smirnov tests. In this manner, distributional assumptions of the models (but not the statistics) are circumvented at the price of associating marginal convergence of k with convergence of the full posterior density.

Brooks and Giudici (2000) propose the monitoring of functionals of parameters which retain their interpretations as the sampler moves between models. The deviance is suggested as a default choice in the absence of superior alternatives. A two-way ANOVA decomposition of the variance of such a functional is formed over multiple chain replications, from which the potential scale reduction factor (PSRF) (Gelman and Rubin, 1992) can be constructed and monitored. Castelloe and Zimmerman (2002) extend this approach firstly to an unbalanced (weighted) two-way ANOVA, to prevent the PRSF being dominated by a few visits to rare models, with the weights being specified in proportion to the frequency of model visits. Castelloe and Zimmerman (2002) also extend their diagnostic to the multivariate (MANOVA) setting on the observation that monitoring several functionals of marginal parameter subsets is more robust than monitoring a single statistic. This general method is clearly reliant on the identification of useful statistics to monitor, but is also sensitive to the extent of approximation induced by violations of the ANOVA assumptions of independence and normality.

Sisson and Fan (2007) propose diagnostics when the underlying model can be formulated in the marked point process framework (Diggle, 1983; Stephens, 2000a). For example, a mixture of an unknown number of univariate normal densities (Equation 3.5) can be represented as a set of k events $\xi_j = (w_j, \mu_j, \sigma_j^2)$, $j = 1, \ldots, k$, in a region $A \subset \mathbb{R}^3$. Given a reference point $v \in A$, in the same space as the events ξ_j (e.g. $v = (\omega, \mu, \sigma^2)$), then the point-to-nearest-event distance, y, is the distance from the point (v) to the nearest event (ξ_j) in A with respect to some distance measure. One can evaluate distributional aspects of the events

$\{\xi_j\}$, through y, as observed from different reference points v. A diagnostic can then be constructed based on comparisons between empirical distribution functions of the distances y, constructed from Markov chain sample paths. Intuitively, as the Markov chains converge, the distribution functions for y constructed from replicate chains should be similar.

This approach permits the direct comparison of full parameter vectors of varying dimension and, as a result, naturally incorporates a measure of across-model convergence. Due to the manner of their construction, Sisson and Fan (2007) are able to monitor an arbitrarily large number of such diagnostics. However, while this approach may have some appeal, it is limited by the need to construct the model in the marked point process setting. Common models which may be formulated in this framework include finite-mixture, change-point and regression models.

Example: Convergence Assessment for Finite Mixture Univariate Normals

We consider the reversible jump sampler of Richardson and Green (1997) implementing a finite mixture of normals model (Equation 3.5) using the enzymatic activity data set (Figure 3.1b). For the purpose of assessing performance of the sampler, we implement five independent sampler replications of length 400,000 iterations.

Figure 3.2a,b illustrates the diagnostic of Brooks et al. (2003b) which provides a test for between-chain convergence based on posterior model probabilities. The pairwise Kolmogorov–Smirnov and χ^2 (all chains simultaneously) tests assume independent realizations. Based on the estimated convergence rate (Brooks et al., 2003b), we retain every 400th iteration to obtain approximate independence. The Kolmogorov–Smirnov statistic cannot reject immediate convergence, with all pairwise chain comparisons well above the critical value of 0.05. The χ^2 statistic cannot reject convergence after the first 10,000 iterations.

Figure 3.2c illustrates the two multivariate PSRFs of Castelloe and Zimmerman (2002) using the deviance as the default statistic to monitor. The solid line shows the ratio of between- and within-chain variation; the broken line indicates the ratio of within-model variation, and the within-model, within-chain variation. The mPSRFs rapidly approach 1, suggesting convergence, beyond 166,000 iterations. This is supported by the independent analysis of Brooks and Giudici (2000) who demonstrate evidence for convergence of this sampler after around 150,000 iterations, although they caution that their chain lengths of only 200,000 iterations were too short for certainty.

Figure 3.2d, adapted from Sisson and Fan (2007), illustrates the PSRF of the distances from each of 100 randomly chosen reference points to the nearest model components, over the five replicate chains. Up to around 100,000 iterations, between-chain variation is still reducing; beyond 300,000 iterations, differences between the chains appear to have stabilized. The intervening iterations mark a gradual transition between these two states. This diagnostic appears to be the most conservative of those presented here.

This example highlights that empirical convergence assessment tools often give varying estimates of when convergence may have been achieved. As a result, it may be prudent to follow the most conservative estimates in practice. While it is undeniable that the benefits for the practitioner in implementing reversible jump sampling schemes are immense, it is arguable that the practical importance of ensuring chain convergence is often overlooked. However, it is also likely that current diagnostic methods are insufficiently advanced to permit a more rigorous default assessment of sampler convergence.

3.3.3 Estimating Bayes Factors

One of the useful by-products of the reversible jump sampler is the ease with which Bayes factors can be estimated. Explicitly expressing marginal or predictive densities of $\mathbf{x}$ under

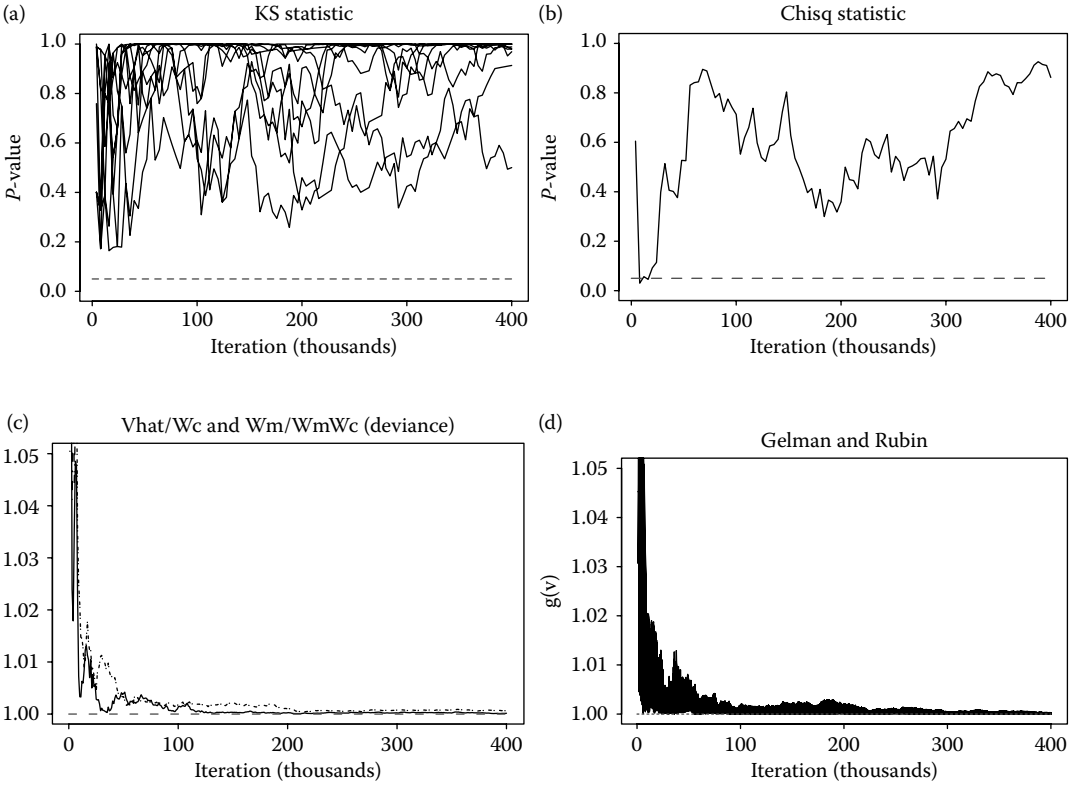

FIGURE 3.2
Convergence assessment for the enzymatic activity data set. (a) Kolmogorov–Smirnov and (b) χ^2 tests of Brooks et al. (2003b). Horizontal line denotes an $\alpha = 0.05$ significance level for test of different sampling distributions. (c) Multivariate PSRFs of Castelloe and Zimmerman (2002) and (d) PSRFvs of Sisson and Fan (2007). Horizontal lines denote the value of each statistic under equal sampling distributions. (From Sisson, S. A. and Fan, Y. 2007. *Statistics and Computing*, 17:357–367. With permission.)

model $\mathcal{M}_k$ as

$$m_k(\mathbf{x}) = \int_{\mathbb{R}^{n_k}} L(\mathbf{x} \mid k, \boldsymbol{\theta}_k) p(\boldsymbol{\theta}_k \mid k)\, d\boldsymbol{\theta}_k,$$

the normalized posterior probability of model $\mathcal{M}_k$ is given by

$$p(k \mid \mathbf{x}) = \frac{p(k) m_k(\mathbf{x})}{\sum_{k' \in \mathcal{K}} p(k') m_{k'}(\mathbf{x})} = \left(1 + \sum_{k' \neq k} \frac{p(k')}{p(k)} B_{k',k}\right)^{-1},$$

where $B_{k',k} = m_{k'}(\mathbf{x})/m_k(\mathbf{x})$ is the Bayes factor of model $\mathcal{M}_{k'}$ to $\mathcal{M}_k$, and $p(k)$ is the prior probability of model $\mathcal{M}_k$. For a discussion of Bayesian model selection techniques, see Chipman et al. (2001), Berger and Pericchi (2001), Kass and Raftery (1995), Ghosh and Samanta (2001), Berger and Pericchi (2004), and Barbieri and Berger (2004). The usual estimator of the posterior model probability, $p(k \mid \mathbf{x})$, is given by the proportion of chain iterations the reversible jump sampler spent in model $\mathcal{M}_k$.

However, when the number of candidate models $|\mathcal{M}|$ is large, the use of reversible jump MCMC algorithms to evaluate Bayes factors raises issues of efficiency. Suppose that model

$\mathcal{M}_k$ accounts for a large proportion of posterior mass. In attempting a between-model move from model $\mathcal{M}_k$, the reversible jump algorithm will tend to persist in this model and visit others models rarely. Consequently, estimates of Bayes factors based on model-visit proportions will tend to be inefficient (Bartolucci and Scaccia, 2003; Han and Carlin, 2001).

Bartolucci et al. (2006) propose enlarging the parameter space of the models under comparison with the same auxiliary variables, $\mathbf{u} \sim q_{d_{k\to k'}}(\mathbf{u})$ and $\mathbf{u}' \sim q_{d_{k'\to k}}(\mathbf{u}')$ (see Equation 3.9), defined under the between-model transitions, so that the enlarged spaces, $(\boldsymbol{\theta}_k, \mathbf{u})$ and $(\boldsymbol{\theta}_{k'}, \mathbf{u}')$, have the same dimension. In this setting, an extension to the bridge estimator for the estimation of the ratio of normalizing constants of two distributions (Meng and Wong, 1996) can be used, by integrating out the auxiliary random process (i.e. $\mathbf{u}$ and $\mathbf{u}'$) involved in the between-model moves. Accordingly, the Bayes factor of model $\mathcal{M}_{k'}$ to $\mathcal{M}_k$ can be estimated using the reversible jump acceptance probabilities as

$$\hat{B}_{k',k} = \frac{\sum_{j=1}^{J_k} \alpha^{(j)}[(k, \boldsymbol{\theta}_k), (k', \boldsymbol{\theta}'_{k'})]/J_k}{\sum_{j=1}^{J_{k'}} \alpha^{(j)}[(k', \boldsymbol{\theta}'_{k'}), (k, \boldsymbol{\theta}_k)]/J_{k'}},$$

where $\alpha^{(j)}[(k, \boldsymbol{\theta}_k), (k', \boldsymbol{\theta}'_{k'})]$ is the acceptance probability (Equation 3.9) of the jth attempt to move from model $\mathcal{M}_k$ to $\mathcal{M}_{k'}$, and where J_k and $J_{k'}$ are the number of proposed moves from model $\mathcal{M}_k$ to $\mathcal{M}_{k'}$ and vice versa during the simulation. Further manipulation is required to estimate $B_{k',k}$ if the sampler does not jump between models $\mathcal{M}_k$ and $\mathcal{M}_{k'}$ directly (Bartolucci et al., 2006). This approach can provide a more efficient way of postprocessing reversible jump MCMC with minimal computational effort.

3.4 Related Multi-Model Sampling Methods

Several alternative multi-model sampling methods are available. Some of these are closely related to the reversible jump MCMC algorithm, or include reversible jump as a special case.

3.4.1 Jump Diffusion

Before the development of the reversible jump sampler, Grenander and Miller (1994) proposed a sampling strategy based on continuous-time jump-diffusion dynamics. This process combines jumps between models at random times, and within-model updates based on a diffusion process according to a Langevin stochastic differential equation indexed by time, t, satisfying

$$d\boldsymbol{\theta}_k^t = dB_k^t + \frac{1}{2}\nabla \log \pi\left(\boldsymbol{\theta}_k^t\right) dt.$$

where dB_k^t denotes an increment of Brownian motion and ∇ the vector of partial derivatives. This method has found some application in signal processing and other Bayesian analyses (Miller et al., 1995; Phillips and Smith, 1996), but has in general been superseded by the more accessible reversible jump sampler. In practice, the continuous-time diffusion must be approximated by a discrete-time simulation. If the time discretization is corrected for via a Metropolis–Hastings acceptance probability, the jump-diffusion sampler actually results in an implementation of reversible jump MCMC (Besag, 1994).

3.4.2 Product Space Formulations

As an alternative to samplers designed for implementation on unions of model spaces, $\boldsymbol{\Theta} = \bigcup_{k\in\mathcal{K}}(\{k\}, \mathbb{R}^{n_k})$, a number of "supermodel" product-space frameworks have been developed, with a state space given by $\boldsymbol{\Theta}^* = \otimes_{k\in\mathcal{K}}(\{k\}, \mathbb{R}^{n_k})$. This setting encompasses all model spaces jointly, so that a sampler needs to simultaneously track $\boldsymbol{\theta}_k$ for all $k \in \mathcal{K}$. The composite parameter vector, $\boldsymbol{\theta}^* \in \boldsymbol{\Theta}^*$, consisting of a concatenation of all parameters under all models, is of fixed dimension, thereby circumventing the necessity of between-model transitions. Clearly, product-space samplers are limited to situations where the dimension of $\boldsymbol{\theta}^*$ is computationally feasible. Carlin and Chib (1995) propose a posterior distribution for the composite model parameter and model indicator given by

$$\pi(k, \boldsymbol{\theta}^* \mid \mathbf{x}) \propto L(\mathbf{x} \mid k, \boldsymbol{\theta}^*_{\mathcal{I}_k}) p(\boldsymbol{\theta}^*_{\mathcal{I}_k} \mid k) p(\boldsymbol{\theta}^*_{\mathcal{I}_{-k}} \mid \boldsymbol{\theta}^*_{\mathcal{I}_k}, k) p(k),$$

where $\mathcal{I}_k$ and $\mathcal{I}_{-k}$ are index sets respectively identifying and excluding the parameters $\boldsymbol{\theta}_k$ from $\boldsymbol{\theta}^*$. Here $\mathcal{I}_k \cap \mathcal{I}_{k'} = \emptyset$ for all $k \neq k'$, so that the parameters for each model are distinct. It is easy to see that the term $p(\boldsymbol{\theta}^*_{\mathcal{I}_{-k}} \mid \boldsymbol{\theta}^*_{\mathcal{I}_k}, k)$, called a "pseudo-prior" by Carlin and Chib (1995), has no effect on the joint posterior $\pi(k, \boldsymbol{\theta}^*_{\mathcal{I}_k} \mid \mathbf{x}) = \pi(k, \boldsymbol{\theta}_k \mid \mathbf{x})$, and its form is usually chosen for convenience. However, poor choices may affect the efficiency of the sampler (Godsill, 2003; Green, 2003).

Godsill (2001) proposes a further generalization of the above by relaxing the restriction that $\mathcal{I}_k \cap \mathcal{I}_{k'} = \emptyset$ for all $k \neq k'$. That is, individual model parameter vectors are permitted to overlap arbitrarily, which is intuitive for, say, nested models. This framework can be shown to encompass the reversible jump algorithm, in addition to the setting of Carlin and Chib (1995). In theory this allows for direct comparison between the three samplers, although this has not yet been fully examined. However, one clear point is that the information contained within $\boldsymbol{\theta}^*_{\mathcal{I}_{-k}}$ would be useful in generating efficient between-model transitions when in model $\mathcal{M}_k$, under a reversible jump sampler. This idea is exploited by Brooks et al. (2003c).

3.4.3 Point Process Formulations

A different perspective on the multi-model sampler is based on spatial birth-and-death processes (Preston, 1977; Ripley, 1977). Stephens (2000a) observed that particular multi-model statistical problems can be represented as continuous-time, marked point processes (Geyer and Møller, 1994). One obvious setting is finite-mixture modeling (Equation 3.5) where the birth and death of mixture components, $\boldsymbol{\phi}_j$, indicate transitions between models. The sampler of Stephens (2000a) may be interpreted as a particular continuous-time, limiting version of a sequence of reversible jump algorithms (Cappé et al., 2003).

A number of illustrative comparisons of the reversible jump, jump diffusion, product space and point process frameworks can be found in the literature. See, for example, Andrieu et al. (2001), Dellaportas et al. (2002), Carlin and Chib (1995), Godsill (2001, 2003), Cappé et al. (2003), and Stephens (2000a).

3.4.4 Multi-Model Optimization

The reversible jump MCMC sampler may be utilized as the underlying random mechanism within a stochastic optimization framework, given its ability to traverse complex spaces efficiently (Andrieu et al., 2000; Brooks et al., 2003a). In a simulated annealing setting, the

sampler would define a stationary distribution proportional to the Boltzmann distribution

$$\mathcal{B}_T(k, \boldsymbol{\theta}_k) \propto \exp\left\{\frac{-f(k, \boldsymbol{\theta}_k)}{T}\right\},$$

where $T \geq 0$ and $f(k, \boldsymbol{\theta}_k)$ is a model-ranking function to be minimized. A stochastic annealing framework will then decrease the value of T according to some schedule while using the reversible jump sampler to explore function space. Assuming adequate chain mixing, as $T \to 0$ the sampler and the Boltzmann distribution will converge to a point mass at $(k^*, \boldsymbol{\theta}^*_{k^*}) = \arg\max f(k, \boldsymbol{\theta}_k)$. Specifications for the model-ranking function may include the Akaike information criterion or Bayesian information criterion (King and Brooks, 2004; Sisson and Fan, 2009), the posterior model probability (Clyde, 1999) or a nonstandard loss function defined on variable-dimensional space (Sisson and Hurn, 2004) for the derivation of Bayes rules.

3.4.5 Population MCMC

The population Markov chain Monte Carlo method (Liang and Wong, 2001; Liu, 2001) may be extended to the reversible jump setting (Jasra et al., 2007). Motivated by simulated annealing (Geyer and Thompson, 1995), N parallel reversible jump samplers are implemented targeting a sequence of related distributions $\{\pi_i\}, i = 1, \ldots, N$, which may be tempered versions of the distribution of interest, $\pi_1 = \pi(k, \boldsymbol{\theta}_k \mid \mathbf{x})$. The chains are allowed to interact, in that the states of any two neighboring (in terms of the tempering parameter) chains may be exchanged, thereby improving the mixing across the population of samplers both within and between models. Jasra et al. (2007) demonstrate superior convergence rates over a single reversible jump sampler. For samplers that make use of tempering or parallel simulation techniques, Gramacy et al. (2010) propose efficient methods of utilizing samples from all distributions (i.e. including those not from π_1) using importance weights, for the calculation of given estimators.

3.4.6 Multi-Model Sequential Monte Carlo

The idea of running multiple samplers over a sequence of related distributions may also considered under a sequential Monte Carlo (SMC) framework (Del Moral et al., 2006). Jasra et al. (2008) propose implementing N separate SMC samplers, each targeting a different subset of model space. At some stage the samplers are allowed to interact and are combined into a single sampler. This approach permits more accurate exploration of models with lower posterior model probabilities than would be possible under a single sampler. As with population MCMC methods, the benefits gained in implementing N samplers must be weighed against the extra computational overheads.

3.5 Discussion and Future Directions

Given the degree of complexity associated with the implementation of reversible jump MCMC, a major focus for future research is in designing simple but efficient samplers, with the ultimate goal of automation. Several authors have provided new insights into the

reversible jump sampler which may contribute toward achieving such goals. For example, Keith et al. (2004) present a generalized Markov sampler, which includes the reversible jump sampler as a special case. Petris and Tardella (2003) demonstrate a geometric approach for sampling from nested models, formulated by drawing from a fixed-dimension auxiliary continuous distribution on the largest model subspace, and then using transformations to recover model-specific samples. Walker (2009) has recently provided a Gibbs sampler alternative to the reversible jump MCMC, using auxiliary variables. Additionally, as noted by Sisson (2005), one does not need to work only with reversible Markov chains, and nonreversible chains may offer opportunities for sampler improvement (Diaconis et al., 2000; Mira and Geyer, 2000; Neal, 2004).

An alternative way of increasing sampler efficiency would be to explore the ideas introduced in adaptive MCMC. As with standard MCMC, any adaptations must be implemented with care—transition kernels dependent on the entire history of the Markov chain can only be used under diminishing adaptation conditions (Haario et al., 2001; Roberts and Rosenthal, 2009). Alternative schemes permit modification of the proposal distribution at regeneration times, when the next state of the Markov chain becomes completely independent of the past (Brockwell and Kadane, 2005; Gilks et al., 1998). Under the reversible jump framework, regeneration can be naturally achieved by incorporating an additional model, from which independent samples can be drawn. Under any adaptive scheme, however, how best to make use of historical chain information remains an open question. Additionally, efficiency gains through adaptations should naturally outweigh the costs of handling chain history and modification of the proposal mechanisms.

Finally, two areas remain underdeveloped in the context of reversible jump simulation. The first of these is perfect simulation, which provides an MCMC framework for producing samples exactly from the target distribution, circumventing convergence issues entirely (Propp and Wilson, 1996). Some tentative steps have been made in this area (Brooks et al., 2006; Møller and Nicholls, 1999). Secondly, while the development of "likelihood-free" MCMC has received much recent attention (Chapter 12, this volume), implementing the sampler in the multi-model setting remains a challenging problem, in terms of both computational efficiency and bias of posterior model probabilities.

Acknowledgments

This work was supported by the Australian Research Council through the Discovery Project scheme (DP0664970 and DP0877432).

References

Al-Awadhi, F., Hurn, M. A., and Jennison, C. 2004. Improving the acceptance rate of reversible jump MCMC proposals. *Statistics and Probability Letters*, 69:189–198.

Andrieu, C., de Freitas, J., and Doucet, A. 2000. Reversible jump MCMC simulated annealing for neural networks. In C. Boutilier and M. Goldszmidt (eds), *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pp. 11–18. Morgan Kaufmann, San Francisco.

Andrieu, C., Djurić, P. M., and Doucet, M. 2001. Model selection by MCMC computation. *Signal Processing*, 81:19–37.

Barbieri, M. M. and Berger, J. O. 2004. Optimal predictive model selection. *Annals of Statistics*, 32:870–897.

Bartolucci, F. and Scaccia, L. 2003. A new approach for estimating the Bayes factor. Technical report, University of Perugia.

Bartolucci, F., Scaccia, L., and Mira, A. 2006. Efficient Bayes factors estimation from reversible jump output. *Biometrika*, 93(1):41–52.

Berger, J. O. and Pericchi, L. R. 2001. Objective Bayesian methods for model selection: Introduction and comparison (with discussion). In P. Lahiri (ed.), *Model Selection*, IMS Lecture Notes—Monograph Series, 38, pp. 135–207. Institute of Mathematical Statistics, Beachwood, OH.

Berger, J. O. and Pericchi, L. R. 2004. Training samples in objective Bayesian model selection. *Annals of Statistics*, 32:841–869.

Besag, J. 1994. Contribution to the discussion of a paper by Grenander and Miller. *Journal of the Royal Statistical Society*, *Series B*, 56:591–592.

Brockwell, A. E. and Kadane, J. B. 2005. Identification of regeneration times in MCMC simulation, with application to adaptive schemes. *Journal of Computational and Graphical Statistics*, 14(2): 436–458.

Brooks, S. P. 1998. Markov chain Monte Carlo method and its application. *The Statistician*, 47: 69–100.

Brooks, S. P. and Giudici, P. 2000. MCMC convergence assessment via two-way ANOVA. *Journal of Computational and Graphical Statistics*, 9:266–285.

Brooks, S. P., Fan, Y., and Rosenthal, J. S. 2006. Perfect forward simulation via simulated tempering. *Communications in Statistics*, 35:683–713.

Brooks, S. P., Friel, N., and King, R. 2003a. Classical model selection via simulated annealing. *Journal of the Royal Statistical Society*, *Series B*, 65:503–520.

Brooks, S. P., Giudici, P., and Philippe, A. 2003b. On non-parametric convergence assessment for MCMC model selection. *Journal of Computational and Graphical Statistics*, 12:1–22.

Brooks, S. P., Guidici, P., and Roberts, G. O. 2003c. Efficient construction of reversible jump Markov chain Monte Carlo proposal distributions (with discusion). *Journal of the Royal Statistical Society*, *Series B*, 65:3–39.

Cappé, O., Robert, C. P., and Rydén, T. 2003. Reversible jump MCMC converging to birth-and-death MCMC and more general continuous time samplers. *Journal of the Royal Statistical Society*, *Series B*, 65:679–700.

Carlin, B. P. and Chib, S. 1995. Bayesian model choice via Markov chain Monte Carlo. *Journal of the Royal Statistical Society*, *Series B*, 57:473–484.

Castelloe, J. M. and Zimmerman, D. L. 2002. Convergence assessment for reversible jump MCMC samplers. Technical Report 313, Department of Statistics and Actuarial Science, University of Iowa.

Celeux, G., Hurn, M. A., and Robert, C. P. 2000. Computational and inferential difficulties with mixture posterior distributions. *Journal of the American Statistical Association*, 95:957–970.

Chipman, H., George, E., and McCulloch, R. E. 2001. The practical implementation of Bayesian model selection (with discussion). In P. Lahiri (ed.), *Model Selection*, IMS Lecture Notes—Monograph Series, 38, pp. 67–134. Institute of Mathematical Statistics, Beachwood, OH.

Clyde, M. A. 1999. Bayesian model averaging and model search strategies. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith (eds), *Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting*, pp. 157–185. Oxford University Press, Oxford.

Cowles, M. K. and Carlin, B. P. 1996. Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91:883–904.

Del Moral, P., Doucet, A., and Jasra, A. 2006. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society*, *Series B*, 68:411–436.

Dellaportas, P. and Papageorgiou, I. 2006. Multivariate mixtures of normals with unknown number of components. *Statistics and Computing*, 16:57–68.

Dellaportas, P., Forster, J. J., and Ntzoufras, I. 2002. On Bayesian model and variable selection using MCMC. *Statistics and Computing*, 12:27–36.

Denison, D. G. T., Mallick, B. K., and Smith, A. F. M. 1998. Automatic Bayesian curve fitting. *Journal of the Royal Statistical Society*, *Series B*, 60:330–350.

Diaconis, P., Holmes, S., and Neal, R. M. 2000. Analysis of a non-reversible Markov chain sampler. *Annals of Applied Probability*, 10:726–752.

Diggle, P. J. 1983. *Statistical Analysis of Spatial Point Patterns*. Academic Press, London.

DiMatteo, I., Genovese, C. R., and Kass, R. E. 2001. Bayesian curve-fitting with free-knot splines. *Biometrika*, 88:1055–1071.

Ehlers, R. S. and Brooks, S. P. 2003. Constructing general efficient proposals for reversible jump MCMC. Technical report, Department of Statistics, Federal University of Paraná.

Ehlers, R. S. and Brooks, S. P. 2008. Adaptive proposal construction for reversible jump MCMC. *Scandinavian Journal of Statistics*, 35:677–690.

Fan, Y. and Brooks, S. P. 2000. Bayesian modelling of prehistoric corbelled domes. *The Statistician*, 49:339–354.

Fan, Y., Peters, G. W., and Sisson, S. A. 2009. Automating and evaluating reversible jump MCMC proposal distributions. *Statistics and Computing*, 19(4):409–421.

Frühwirth-Schnatter, S. 2001. Markov chain Monte Carlo estimation of classical and dynamic switching and mixture models. *Journal of the American Statistical Association*, 96:194–209.

Gelman, A. and Rubin, D. B. 1992. Inference from iterative simulations using multiple sequences. *Statistical Science*, 7:457–511.

George, A. W., Mengersen, K. L., and Davis, G. P. 1999. A Bayesian approach to ordering gene markers. *Biometrics*, 55:419–429.

George, E. I. and McCulloch, R. E. 1993. Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88:881–889.

Geyer, C. J. and Møller, J. 1994. Simulation procedures and likelihood inference for spatial point processes. *Scandinavian Journal of Statistics*, 21:359–373.

Geyer, C. J. and Thompson, E. A. 1995. Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90:909–920.

Ghosh, J. K. and Samanta, T. 2001. Model selection: An overview. *Current Science*, 80:1135–1144.

Gilks, W. R., Roberts, G. O., and Sahu, S. K. 1998. Adaptive Markov chain Monte Carlo through regeneration. *Journal of the American Statistical Association*, 93:1045–1054.

Godsill, S. 2001. On the relationship between Markov chain Monte Carlo methods for model uncertainty. *Journal of Computational and Graphical Statistics*, 10:1–19.

Godsill, S. 2003. Discussion of Trans-dimensional Markov chain Monte Carlo by P. J. Green. In P. J. Green, N. L. Hjort, and S. Richardson (eds), *Highly Structured Stochastic Systems*, pp. 199–203. Oxford University Press, Oxford.

Gramacy, R. B., Samworth, R. J., and King, R. 2010. Importance tempering. *Statistics and Computing*, 20:1–7.

Green, P. J. 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732.

Green, P. J. 2001. A primer on Markov chain Monte Carlo. In O. E. Barndorff-Nielsen, D. R. Cox, and C. Klüppelberg (eds), *Complex Stochastic Systems*, Monographs on Statistics and Probability, 87, pp. 1–62. Chapman & Hall/CRC, Boca Raton, FL.

Green, P. J. 2003. Trans-dimensional Markov chain Monte Carlo. In P. J. Green, N. L. Hjort, and S. Richardson (eds), *Highly Structured Stochastic Systems*, pp. 179–198. Oxford University Press, Oxford.

Green, P. J. and Mira, A. 2001. Delayed rejection in reversible jump Metropolis-Hastings. *Biometrika*, 88:1035–1053.

Grenander, U. and Miller, M. I. 1994. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society*, *Series B*, 56:549–603.

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

Han, C. and Carlin, B. P. 2001. MCMC methods for computing Bayes factors: A comparative review. *Journal of the American Statistical Association*, 96:1122–1132.

Hastie, D. 2004. Developments in Markov chain Monte Carlo. PhD thesis, University of Bristol.

Hastie, T. J. and Tibshirani, R. J. 1990. *Generalised Additive Models*. Chapman & Hall, London.

Hastings, W. K. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:59–109.

Hurn, M., Justel, A., and Robert, C. P. 2003. Estimating mixtures of regressions. *Journal of Computational and Graphical Statistics*, 12:55–79.

Jasra, A., Doucet, A., Stephens, D. A., and Holmes, C. 2008. Interacting sequential Monte Carlo samplers for trans-dimensional simulation. *Computational Statistics and Data Analysis*, 52(4): 1765–1791.

Jasra, A., Holmes, C., and Stephens, D. A. 2005. MCMC methods and the label switching problem. *Statistical Science*, 20(1):50–67.

Jasra, A., Stephens, D. A., and Holmes, C. C. 2007. Population-based reversible jump Markov chain Monte Carlo. *Biometrika*, 94:787–807.

Kass, R. E. and Raftery, A. E. 1995. Bayes factors. *Journal of the American Statistical Association*, 90:773–796.

Keith, J. M., Kroese, D. P., and Bryant, D. 2004. A generalized Markov sampler. *Methodology and Computing in Applied Probability*, 6:29–53.

King, R. and Brooks, S. P. 2004. A classical study of catch-effort models for Hector's dolphins. *Journal of the American Statistical Association*, 99:325–333.

Kirkpatrick, S. 1984. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34:975–986.

Lawler, G. and Sokal, A. 1988. Bounds on the L^2 spectrum for Markov chains and Markov processes. *Transactions of the American Mathematical Society*, 309:557–580.

Liang, F. and Wong, W. H. 2001. Real parameter evolutionary Monte Carlo with applications to Bayesian mixture models. *Journal of the American Statistical Association*, 96:653–666.

Liu, J. S. 2001. *Monte Carlo Strategies in Scientific Computing*. Springer, New York.

Liu, J. S., Liang, F., and Wong, W. H. 2001. A theory for dynamic weighing in Monte Carlo computation. *Journal of the American Statistical Association*, 96(454):561–573.

Meng, X. L. and Wong, W. H. 1996. Simulating ratios of normalising constants via a simple identity: A theoretical exploration. *Statistica Sinica*, 6:831–860.

Mengersen, K. L., Robert, C. P., and Guihenneuc-Joyaux, C. 1999. MCMC convergence diagnostics: A review. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith (eds), *Bayesian Statistics 6*, pp. 415–140. Oxford University Press, Oxford.

Miller, M. I., Srivastava, A., and Grenander, U. 1995. Conditional-mean estimation via jump-diffusion processes in multiple target tracking/recognition. *IEEE Transactions on Signal Processing*, 43:2678–2690.

Mira, A. and Geyer, C. J. 2000. On non-reversible Markov chains. In N. Madras (ed.), *Monte Carlo Methods*, pp. 93–108. American Mathematical Society, Providence, RI.

Møller, J. and Nicholls, G. K. 1999. Perfect simulation for sample-based inference. Technical report, Aalborg University.

Neal, R. M. 2004. Improving asymptotic variance of MCMC estimators: Non-reversible chains are better. Technical Report 0406, Department of Statisics, University of Toronto.

Nott, D. J. and Green, P. J. 2004. Bayesian variable selection and the Swendsen-Wang algorithm. *Journal of Computational and Graphical Statistics*, 13(1):141–157.

Nott, D. J. and Leonte, D. 2004. Sampling schemes for Bayesian variable selection in generalised linear models. *Journal of Computational and Graphical Statistics*, 13(2):362–382.

Ntzoufras, I., Dellaportas, P., and Forster, J. J. 2003. Bayesian variable and link determination for generalised linear models. *Journal of Statistical Planning and Inference*, 111:165–180.

Papathomas, M., Dellaportas, P., and Vasdekis, V. G. S. 2009. A general proposal construction for reversible jump MCMC. Technical report, Athens University of Economics and Business.

Peskun, P. 1973. Optimum Monte Carlo sampling using Markov chains. *Biometrika*, 60:607–612.

Petris, G. and Tardella, L. 2003. A geometric approach to transdimensional Markov chain Monte Carlo. *Canadian Journal of Statistics*, 31.

Phillips, D. B. and Smith, A. F. M. 1996. Bayesian model comparison via jump diffusions. In W. R. Gilks, S. Richardson, and D. J, Spiegelhalter (eds), *Markov Chain Monte Carlo in Practice*, pp. 215–239. Chapman & Hall, London.

Preston, C. J. 1977. Spatial birth-and-death processes. *Bulletin of the International Statistical Institute*, 46:371–391.

Propp, J. G. and Wilson, D. B. 1996. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9:223–252.

Richardson, S. and Green, P. J. 1997. On Bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society*, *Series B*, 59: 731–792.

Ripley, B. D. 1977. Modelling spatial patterns (with discussion). *Journal of the Royal Statistical Society*, *Series B*, 39:172–212.

Roberts, G. O. 2003. Linking theory and practice of MCMC. In P. J. Green, N. Hjort, and S. Richardson (eds), *Highly Structured Stochastic Systems*, pp. 145–166. Oxford University Press.

Roberts, G. O. and Rosenthal, J. S. 2009. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18:349–367.

Sisson, S. A. 2005. Trans-dimensional Markov chains: A decade of progress and future perspectives. *Journal of the American Statistical Association*, 100:1077–1089.

Sisson, S. A. and Fan, Y. 2007. A distance-based diagnostic for trans-dimensional Markov chains. *Statistics and Computing*, 17:357–367.

Sisson, S. A. and Fan, Y. 2009. Towards automating model selection for a mark-recapture-recovery analysis. *Applied Statistics*, 58(2):247–266.

Sisson, S. A. and Hurn, M. A. 2004. Bayesian point estimation of quantitative trait loci. *Biometrics*, 60:60–68.

Smith, M. and Kohn, R. 1996. Nonparametric regression using Bayesian variable selection. *Journal of Econometrics*, 75:317–344.

Stephens, M. 2000a. Bayesian analysis of mixture models with an unknown number of components—an alternative to reversible jump methods. *Annals of Statistics*, 28:40–74.

Stephens, M. 2000b. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society*, *Series B*, 62:795–809.

Tadesse, M., Sha, N., and Vannucci, M. 2005. Bayesian variable selection in clustering high-dimensional data. *Journal of the American Statistical Association*, 100:602–617.

Tierney, L. 1998. A note on Metropolis–Hastings kernels for general state spaces. *Annals of Applied Probability*, 8:1–9.

Tierney, L. and Mira, A. 1999. Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in Medicine*, 18:2507–2515.

Trias, M., Vecchio, A., and Vetich, J. 2009. Delayed rejection schemes for efficient Markov chain Monte Carlo sampling of multimodal distributions. Technical report, Universitat de les Illes Balears.

Vermaak, J., Andrieu, C., Doucet, A., and Godsill, S. J. 2004. Reversible jump Markov chain Monte Carlo strategies for Bayesian model selection in autoregressive processes. *Journal of Time Series Analysis*, 25(6):785–809.

Walker, S. G. 2009. A Gibbs sampling alternative to reversible jump MCMC. Technical report, University of Kent.

4

Optimal Proposal Distributions and Adaptive MCMC

Jeffrey S. Rosenthal

4.1 Introduction

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

4.1.1 The Metropolis–Hastings Algorithm

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

$$\alpha(\mathbf{x}, \mathbf{y}) = \begin{cases} \min\left\{ \dfrac{\pi(\mathbf{y})}{\pi(\mathbf{x})} \dfrac{q(\mathbf{y}, \mathbf{x})}{q(\mathbf{x}, \mathbf{y})}, 1 \right\}, & \pi(\mathbf{x})\, q(\mathbf{x}, \mathbf{y}) > 0, \\ 1, & \pi(\mathbf{x})\, q(\mathbf{x}, \mathbf{y}) = 0. \end{cases} \tag{4.1}$$

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

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

4.1.2 Optimal Scaling

It has long been recognized that the choice of the proposal density $q(\mathbf{x}, \mathbf{y})$ is crucial to the success (e.g. rapid convergence) of the Metropolis–Hastings algorithm. Of course, the

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

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

In recent years, significant progress has been made in identifying optimal proposal scalings, in terms of such tangible values as asymptotic acceptance rate. Under certain conditions, these results can describe the optimal scaling precisely. These issues are discussed in Section 4.2 below.

4.1.3 Adaptive MCMC

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

Suppose $\{P_\gamma\}_{\gamma \in \mathcal{Y}}$ is a family of Markov chains, each having stationary distribution π. (For example, perhaps P_γ corresponds to an RWM algorithm with increment distribution $N(0, \gamma^2 I_d)$.) An adaptive MCMC algorithm would randomly update the value of γ at each iteration, in an attempt to find the best value. Adaptive MCMC has been applied in a variety of contexts (e.g. Haario et al., 2001; Giordani and Kohn, 2006; Roberts and Rosenthal, 2009), including problems in statistical genetics (Turro et al., 2007).

Counterintuitively, adaptive MCMC algorithms may not always preserve the stationarity of π. However, if the adaptations are designed to satisfy certain conditions, then stationarity is guaranteed, and significant speed-ups are possible. These issues are discussed in Section 4.3 below.

4.1.4 Comparing Markov Chains

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

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

Alternatively, P_1 *has smaller variance than* P_2 if $\mathbf{Var}\big(\frac{1}{n}\sum_{i=1}^n g(X_i)\big)$ is smaller when $\{X_i\}$ follows P_1 than when it follows P_2. This definition concerns the variance of a functional g, and may depend on which g is chosen, and also perhaps on n and/or the starting distribution.

Usually we assume that the Markov chain $\{X_n\}$ is in stationarity, so $\Pr(X_i \in A) = \pi(A)$, and $\Pr(X_{i+1} \in A \mid X_i = x) = P(x, A)$ where P is the Markov chain kernel being followed.

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

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

Such competing definitions of "better" Markov chain mean that the optimal choice of MCMC may depend on the specific question being asked. However, we will see in Section 4.2 that in some circumstances these different definitions are all equivalent, leading to uniformly optimal choices of algorithm parameters.

4.2 Optimal Scaling of Random-Walk Metropolis

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

4.2.1 Basic Principles

A first observation is that if σ is very small, then virtually all proposed moves will be accepted, but they will represent very small movements, so overall the chain will not mix well (Figure 4.1). Similarly, if σ is very large, then most moves will be rejected, so the chain will usually not move at all (Figure 4.2). What is needed is a value of σ between the two extremes, thus allowing for reasonable-sized proposal moves together with a reasonably high acceptance probability (Figure 4.3).

A simple way to avoid the extremes is to monitor the *acceptance rate* of the algorithm, that is, the fraction of proposed moves which are accepted. If this fraction is very close to 1, this suggests that σ is too small (as in Figure 4.1). If this fraction is very close to 0, this suggests that σ is too large (as in Figure 4.2). But if this fraction is far from 0 and far from 1, then we have managed to avoid both extremes (Figure 4.3).

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

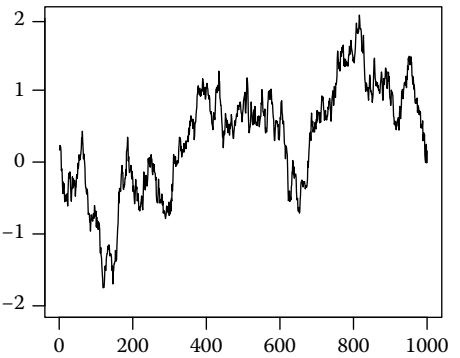

FIGURE 4.1
Trace plot with small σ, large acceptance rate, and poor mixing.

4.2.2 Optimal Acceptance Rate as $d \to \infty$

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

$$\pi(x_1, x_2, \ldots, x_d) = f(x_1) f(x_2) \ldots f(x_d), \tag{4.2}$$

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

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

More precisely, their result is the following. Suppose that $\sigma = \ell/\sqrt{d}$ for some $\ell > 0$. Then as $d \to \infty$, if time is speeded up by a factor of d, and space is shrunk by a factor of $\sqrt{d}$, then each component of the Markov chain converges to a diffusion having stationary distribution f, and speed function given by $h(\ell) = 2\,\ell^2\,\Phi\left(-\sqrt{I}\ell/2\right)$, where Φ is the cumulative distribution

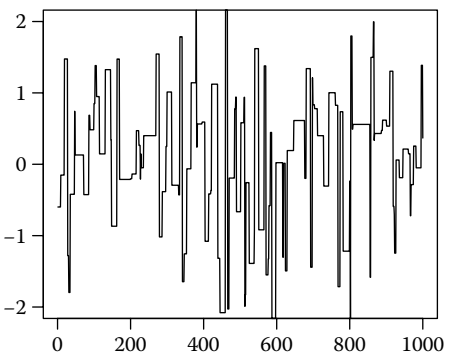

FIGURE 4.2
Trace plot with large σ, small acceptance rate, and poor mixing.

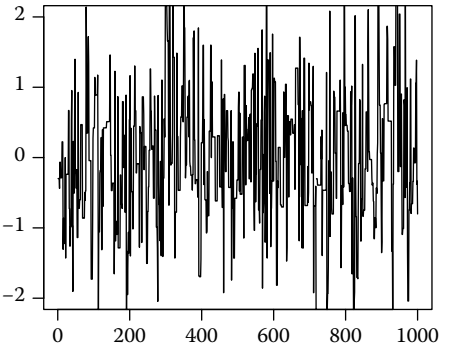

FIGURE 4.3
Trace plot with medium σ, medium acceptance rate, and good mixing.

function of a standard normal, and I is a constant depending on f, given in fact by $I = \int_{-\infty}^{\infty} \left[\left(\frac{f'(x)}{f(x)} \right)^2 \right] f(x)\, dx$.

It follows that this diffusion is optimized (in terms of *any* of the criteria of Section 4.1.4) when ℓ is chosen to maximize $h(\ell)$. It is computed numerically that this optimal value of ℓ is given by $\ell_{\text{opt}} \doteq 2.38/\sqrt{I}$.

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

Figure 4.4 plots $h(\ell)$ versus ℓ, and Figure 4.5 plots $h(\ell)$ versus $A(\ell)$. (We take $I = 1$ for definiteness, but any other value of I would simply multiply all the values by a constant.) In particular, the relative speed $h(\ell)$ remains fairly close to its maximum as long as ℓ is within, say, a factor of 2 of its optimal value. Equivalently, the algorithm remains relatively efficient as long as the asymptotic acceptance rate $A(\ell)$ is between, say, 0.1 and 0.6.

Of course, the above results are all asymptotic as $d \to \infty$. Numerical studies (e.g. Gelman et al., 1996; Roberts and Rosenthal, 2001) indicate that the limiting results do seem to well approximate finite-dimensional situations for d as small as 5. On the other hand, they

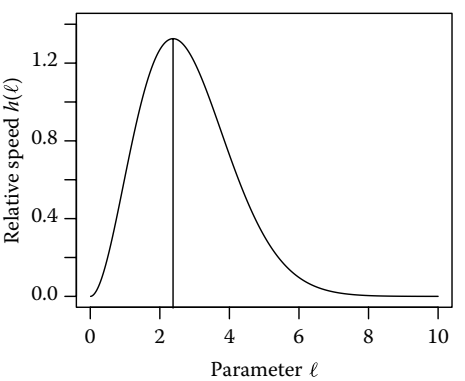

FIGURE 4.4
Algorithm relative speed $h(\ell)$ as a function of the parameter ℓ.

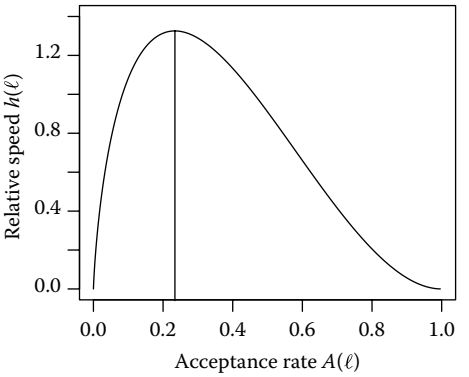

FIGURE 4.5
Algorithm relative speed $h(\ell)$ as a function of acceptance rate $A(\ell)$.

do not apply to one-dimensional increments, for example; numerical studies on normal distributions show that when $d = 1$, the optimal acceptance rate is approximately 0.44. Finally, these results are all on continuous spaces, but there have also been studies of optimal scaling for discrete Metropolis algorithms (Neal et al., 2007).

4.2.3 Inhomogeneous Target Distributions

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

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

$$\pi(\mathbf{x}) = \prod_{i=1}^{d} C_i f(C_i x_i), \tag{4.3}$$

where the $\{C_i\}$ are themselves i.i.d. from some fixed distribution. (Thus, Equation 4.2 corresponds to the special case where the C_i are constant.) They proved that in this case, the result of Roberts et al. (1997) still holds (including the optimal acceptance rate of 0.234), except that the limiting diffusion speed is divided by an "inhomogeneity factor" of $b \equiv \mathrm{E}(C_i^2)/(\mathrm{E}(C_i))^2 \geq 1$. In particular, the more inhomogeneous the target distribution (i.e. the greater the variability of the C_i), the slower the resulting algorithm.

As a special case, if the target distribution is $N(0, \Sigma)$ for some d-dimensional covariance matrix Σ, and the increment distribution is of the form $N(0, \Sigma_p)$, then by change of basis this is equivalent to the case of proposal increment $N(0, I_d)$ and target distribution $N(0, \Sigma\,\Sigma_p^{-1})$. In the corresponding eigenbasis, this target distribution is of the form (Equation 4.3) where now $C_i = \sqrt{\lambda_i}$, with $\{\lambda_i\}_{i=1}^{d}$ the eigenvalues of the matrix $\Sigma\,\Sigma_p^{-1}$. For large d, this approximately corresponds to the case where the $\{C_i\}$ are random with $\mathrm{E}(C_i) = \frac{1}{d}\sum_{j=1}^{d}\sqrt{\lambda_j}$ and $\mathrm{E}(C_i^2) = \frac{1}{d}\sum_{j=1}^{d}\lambda_j$. The inhomogeneity factor b then becomes

$$b \equiv \frac{\mathrm{E}(C_i^2)}{(\mathrm{E}(C_i))^2} \approx \frac{\frac{1}{d}\sum_{j=1}^{d}\lambda_j}{\left(\frac{1}{d}\sum_{j=1}^{d}\sqrt{\lambda_j}\right)^2} = d\,\frac{\sum_{j=1}^{d}\lambda_j}{\left(\sum_{j=1}^{d}\sqrt{\lambda_j}\right)^2}, \tag{4.4}$$

with $\{\lambda_j\}$ the eigenvalues of $\Sigma\,\Sigma_p^{-1}$. This expression is maximized when the $\{\lambda_j\}$ are constant, that is, when $\Sigma\,\Sigma_p^{-1}$ is a multiple of the identity, or in other words, when Σ_p is proportional to Σ.

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

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

$$\Sigma_p = \left[\frac{(2.38)^2}{d}\right]\Sigma. \tag{4.5}$$

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

4.2.4 Metropolis-Adjusted Langevin Algorithm

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

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

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

Roberts and Rosenthal (1998) proved that in this case, under the same i.i.d. target assumption (Equation 4.2), a similar optimal scaling result holds. This time the scaling is $\sigma = \ell/d^{1/6}$ (as opposed to $\ell/\sqrt{d}$), and the optimal value ℓ_{opt} has the optimal asymptotic acceptance rate $A(\ell_{\text{opt}}) = 0.574$ (as opposed to 0.234).

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

4.2.5 Numerical Examples

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

4.2.5.1 Off-Diagonal Covariance

Let M be the $d \times d$ matrix having diagonal elements 1, and off-diagonal elements given by the product of the row and column number divided by d^2, that is, $m_{ii} = 1$, and $m_{ij} = ij/d^2$ for $j \neq i$. Then let $\Sigma^{-1} = M^2$ (since M is symmetric, Σ is positive-definite), and let the target density π be that of $N(0, \Sigma)$. (Equivalently, π is such that $\mathbf{X} \sim \pi$ if $\mathbf{X} = MZ$, where Z is a 10-tuple of i.i.d. univariate standard normals.)

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

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

σ	Mean Acc. Rate	Estimate	RMSE
0.1	0.836	0.992 ± 0.066	0.074
0.7	0.230	1.032 ± 0.019	0.018
3.0	0.002	1.000 ± 0.083	0.085

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

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

4.2.5.2 Inhomogeneous Covariance

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

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

σ	Mean Acc. Rate	Estimate	RMSE
0.7	0.230	114.8 ± 28.2	30.5

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

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

σ	Mean Acc. Rate	Estimate	RMSE
0.7	0.294	100.25 ± 1.91	1.83

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

Of course, in general the target covariance matrix will not be known, and it is not at all clear (especially in high dimensions) how one could arrange for proposal increment covariances to mimic the target covariance. One promising solution is adaptive MCMC, discussed in the next section. In particular, Section 4.3.2 considers the adaptive Metropolis algorithm and shows how it can successfully mimic the target covariance without any *a priori* knowledge about it, even in hundreds of dimensions.

4.2.6 Frequently Asked Questions

Isn't a larger acceptance rate always preferable?

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

Is it essential that the acceptance rate be exactly 0.234?

No. As shown in Figure 4.5, the algorithm's efficiency remains high whenever the acceptance rate is between about 0.1 and 0.6.

Are these asymptotic results relevant to finite-dimensional problems?

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

Do these results hold for all target distributions?

No. They are only proved for very special cases involving independent target components. However, within that class they appear to be fairly robust (albeit sometimes with an even *lower* optimal acceptance rate than 0.234), and simulations seem to suggest that they approximately hold in other cases too. Furthermore, by change of basis, the results apply to all

normal target distributions, too. And the general principle that the scaling should be neither too large nor too small applies much more generally, to virtually all "local" MCMC algorithms.

Do these results hold for multimodal distributions?

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

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

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

Doesn't optimality depend on which criterion is used?

Yes, in general, but these asymptotic diffusion results are valid for *any* optimality measure. That is because in the limit the processes each represent precisely the same *diffusion*, just scaled with a different speed factor. So, running a suboptimal algorithm for n steps is precisely equivalent (in the limit) to running the optimal algorithm for m steps, where $m < n$. In other words, with a suboptimal algorithm you have to run for longer to achieve precisely the same result, which is less efficient by any sensible efficiency measure at all, including all of those in Section 4.1.4.

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

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

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

No, a smaller scaling would correspond to letting $\ell \to 0$, while a larger scaling would correspond to letting $\ell \to \infty$, either of which would lead to an asymptotically zero-efficiency algorithm (*cf.* Figure 4.5). The $O(d^{-1/2})$ or $O(d^{-1/6})$ scaling is the only one that leads to a nonzero limit, and thus the only scaling leading to optimality as $d \to \infty$.

4.3 Adaptive MCMC

Even if we have some idea of what criteria make an MCMC algorithm optimal, this still leaves the question of how to *find* this optimum, that is, how to run a Markov chain with

(approximately) optimal characteristics. For example, even if we are convinced that an acceptance rate of 0.234 is optimal, how do we find the appropriate proposal scaling to achieve this?

One method, commonly used, is trial and error: if the acceptance rate seems too high, then we reduce the proposal scaling σ and try again (or if it seems too low, then we increase the scaling). This method is often successful, but it is generally time-consuming, requiring repeated manual intervention by the user. Furthermore, such a method cannot hope to find more complicated improvements, for example making the proposal covariance matrix Σ_p approximately proportional to the (unknown) target covariance matrix Σ as in Equation 4.5 (which requires choosing $d(d-1)/2$ separate covariance matrix entries). It is possible to use more refined versions of this, for example with increasing trial run lengths to efficiently zero in on good proposal scale and shape values (Pasarica and Gelman, 2010), but this is still not sufficient in difficult high-dimensional problems.

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

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

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

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

4.3.1 Ergodicity of Adaptive MCMC

One might think that, as long as each individual Markov chain P_γ converges to π, any adaptive mixture of the chains must also converge to π. However, this is not the case. For a simple counterexample (illustrated interactively by Rosenthal, 2004; see also Atchadé and Rosenthal, 2005; Roberts and Rosenthal, 2007), let $\mathcal{Y} = \{1, 2\}$, let $\mathcal{X} = \{1, 2, 3, 4\}$, let $\pi(1) = \pi(3) = \pi(4) = 0.333$ and $\pi(2) = 0.001$. Let each P_γ be an RWM algorithm, with proposal $Y_{n+1} \sim U\{X_n - 1, X_n + 1\}$ for P_1, or $Y_{n+1} \sim U\{X_n - 2, X_n - 1, X_n + 1, X_n + 2\}$ for P_2. (Of course, any proposed moves out of $\mathcal{X}$ are always rejected, i.e. $\pi(x) = 0$ for $x \notin \mathcal{X}$.) Define the adaptation by saying that $\Gamma_{n+1} = 2$ if the nth proposal was accepted, otherwise $\Gamma_{n+1} = 1$. Then each P_γ is reversible with respect to π. However, the adaptive algorithm can get "stuck" with $X_n = \Gamma_n = 1$ for long stretches (and only escape with probability 0.001/0.333), so the limiting distribution of X_n is weighted too heavily toward 1 (and too lightly toward 3 and 4).

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

In particular, Roberts and Rosenthal (2007) proved that $\lim_{n\to\infty} \sup_{A\subseteq\mathcal{X}} \|\Pr(X_n \in A) - \pi(A)\| = 0$ (asymptotic convergence), and also $\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} g(X_i) = \pi(g)$ for all bounded $g : \mathcal{X} \to \mathbf{R}$ (WLLN), assuming only the *diminishing* (a.k.a. *vanishing*) *adaptation* condition

$$\lim_{n\to\infty} \sup_{x\in\mathcal{X}} \|P_{\Gamma_{n+1}}(x,\cdot) - P_{\Gamma_n}(x,\cdot)\| = 0 \quad \text{in probability,} \tag{4.6}$$

and also the *containment* (a.k.a. *bounded convergence*) condition

$$\{M_\epsilon(X_n, \Gamma_n)\}_{n=0}^{\infty} \text{ is bounded in probability,} \quad \epsilon > 0, \tag{4.7}$$

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

Now, Equation 4.7 is a technical condition which is satisfied for virtually all reasonable adaptive schemes. For example, it holds whenever $\mathcal{X} \times \mathcal{Y}$ is finite, or is compact in some topology in which either the transition kernels P_γ, or the Metropolis–Hastings proposal kernels Q_γ, have jointly continuous densities. It also holds for adaptive RWM and Metropolis-within-Gibbs algorithms under very general conditions (Bai et al., 2008). (It is, however, possible to construct pathological counterexamples, where containment does not hold; see Yang, 2008b and Bai et al., 2008.) So, in practice, the requirement (Equation 4.7) can be largely ignored.

By contrast, condition (Equation 4.6) is more fundamental. It requires that the amount of adapting at the nth iteration goes to 0 as $n \to \infty$. (Note that the *sum* of the adaptations can still be infinite, i.e. an infinite total amount of adaptation is still permissible, and it is not necessarily required that the adaptive parameters $\{\Gamma_n\}$ converge to some fixed value.) Since the user can choose the adaptive updating scheme, Equation 4.6 can be ensured directly through careful planning. For example, if the algorithm adapts at the nth iteration only with probability $p(n)$, then Equation 4.6 is automatically satisfied if $p(n) \to 0$. Alternatively, if the choice of γ depends on an empirical average over iterations 1 through n, then the influence of the nth iteration is just $O(1/n)$ and hence goes to 0.

Such results allow us to update our parameters $\{\Gamma_n\}$ in virtually any manner we wish, so long as (Equation 4.6) holds. So, what adaptations are beneficial?

4.3.2 Adaptive Metropolis

The first important modern use of adaptive MCMC was the adaptive Metropolis (AM) algorithm of Haario et al. (2001). This algorithm is motivated by the observation (Equation 4.5) that, for RWM in $\mathbf{R}^d$, at least with normal target distributions, it is optimal to have a proposal covariance matrix of the form $(2.38)^2/d$ times the target covariance matrix Σ. Since Σ is in general unknown, it is estimated by Σ_n, the empirical covariance matrix of $X_0, \ldots, X_n$.

Thus, the AM algorithm essentially uses a proposal distribution for the nth iteration given by

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

To ensure that the proposal covariances do not simply collapse to 0 (which could violate (Equation 4.7)), Haario et al. (2001) added $\epsilon\, I_d$ to Σ_n at each iteration, for some small

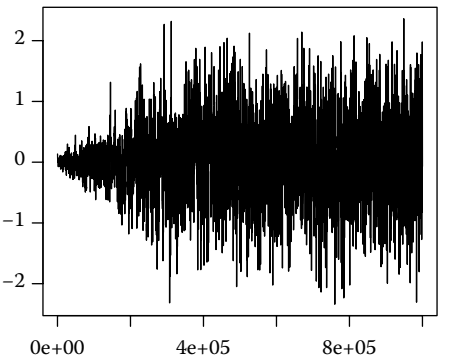

FIGURE 4.6
Trace plot of first coordinate of AM in dimension 100.

$\epsilon > 0$. Another possibility (Roberts and Rosenthal, 2009) is to instead let the proposal be a mixture distribution of the form

$$(1-\beta)N\big(X_n, \left[\frac{(2.38)^2}{d}\right]\Sigma_n\big) + \beta N\big(X_n, \Sigma_0\big)$$

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

Since empirical estimates change at the nth iteration by only $O(1/n)$, it follows that the diminishing adaptation condition (Equation 4.6) will be satisfied. Furthermore, the containment condition (Equation 4.7) will certainly be satisfied if one restricts to compact regions (Haario et al., 2001; Roberts and Rosenthal, 2009), and in fact containment still holds provided the target density π decays at least polynomially in each coordinate, a very mild assumption (Bai et al., 2008). So, AM is indeed a valid sampling algorithm.

Computer simulations (Roberts and Rosenthal, 2009) demonstrate that this AM algorithm will indeed "learn" the target covariance matrix, and approach an optimal algorithm, even in very high dimensions. While it may take many iterations before the adaptation significantly improves the algorithm, in the end it will converge considerably faster than a nonadapted RWM algorithm. For an AM run in dimension $d = 100$ (where the target was a normal distribution with an irregular and highly skewed covariance matrix), Figure 4.6 shows a trace plot of the first coordinate and Figure 4.7 a graph of the inhomogeneity factor b in Equation 4.4. These figures show that the run initially underestimates the variability of the first coordinate, which would lead to drastically incorrect estimates. However, after about 250,000 iterations, the algorithm has "found" a good proposal increment covariance matrix, so that b gets close to 1, and the trace plot correctly finds the true variability of the first coordinate. Such adaptation could never have been done manually, because of the large dimension, but the computer eventually finds a good algorithm. This shows the potential of adaptive MCMC to find good algorithms that cannot be found by hand.

4.3.3 Adaptive Metropolis-within-Gibbs

A standard alternative to the usual full-dimensional Metropolis algorithm is the "Metropolis-within-Gibbs" algorithm (arguably a misnomer, since the original work of

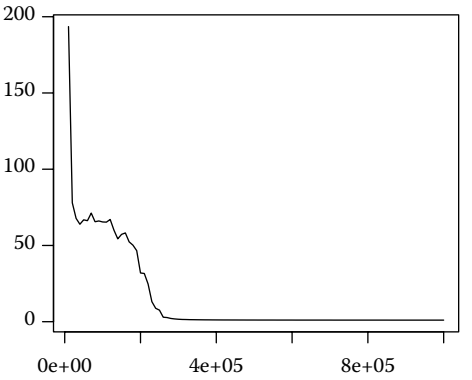

FIGURE 4.7
Trace plot of inhomogeneity factor b for AM in dimension 100.

Metropolis et al., 1953, corresponded to what we now call Metropolis-within-Gibbs). Here the variables are updated one at a time (in either systematic or random order), each using a Metropolis algorithm with a one-dimensional proposal.

To be specific, suppose that the ith coordinate is updated using a proposal increment distribution $N(0, e^{2ls_i})$, so ls_i is the log of the standard deviation of the increment. Obviously, we would like to find optimal values of the ls_i, which may of course be different for the different variables. We even have a rule of thumb from the end of Section 4.2.3, that each ls_i should be chosen so that the acceptance rate is approximately 0.44. However, even with this information, it is very difficult (if not impossible) in high dimensions to optimize each ls_i manually. Instead, an adaptive algorithm might be used.

One way (Roberts and Rosenthal, 2009) to adapt the ls_i values is to break up the run into "batches" of, say, 50 iterations each. After the nth batch, we update each ls_i by adding or subtracting an adaptation amount $\delta(n)$. The adapting attempts to make the acceptance rate of proposals for variable i as close as possible to 0.44. Specifically, we increase ls_i by $\delta(n)$ if the fraction of acceptances of variable i was more than 0.44 on the nth batch, or decrease ls_i by $\delta(n)$ if it was less. (A related componentwise adaptive scaling method, a one-dimensional analog of the original AM algorithm of Haario et al., 2001, is presented in Haario et al., 2005.)

To satisfy condition (Equation 4.6) we require $\delta(n) \to 0$; for example, we might take $\delta(n) = \min(0.01, n^{-1/2})$. As for Equation 4.7, it is easily seen to be satisfied if we restrict each ls_i to a finite interval $[-M, M]$. However, even this is not necessary, since it is proved by Bai et al. (2008) that Equation 4.7 is always satisfied for this algorithm, provided only that the target density π decreases at least polynomially in each direction (a very mild condition). Hence, the restriction (Equation 4.7) is once again not of practical concern.

Simulations (Roberts and Rosenthal, 2009) indicate that this adaptive Metropolis-within-Gibbs algorithm does a good job of correctly scaling the ls_i values, even in dimensions as high as 500, leading to chains which mix much faster than those with pre-chosen proposal scalings. The algorithm has recently been applied successfully to high-dimensional inference for statistical genetics (Turro et al., 2007). We believe it will be applied to many more sampling problems in the near future. Preliminary general-purpose software to implement this algorithm is now available (Rosenthal, 2007).

4.3.4 State-Dependent Proposal Scalings

Another approach involves letting the proposal scaling depend on the current state X_n, so that, for example, given $X_n = x$, we might propose $Y_{n+1} \sim N(x, \sigma_x^2)$. In this case, the acceptance probability (Equation 4.1) becomes

$$\alpha(x, y) = \min\left[1, \frac{\pi(y)}{\pi(x)}\left(\frac{\sigma_x}{\sigma_y}\right)^d \exp\left(-\frac{1}{2}(x-y)^2(\sigma_y^{-2} - \sigma_x^{-2})\right)\right]. \tag{4.8}$$

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

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

Once again, condition (Equation 4.6) is satisfied provided $\delta(n) \to 0$, and (Equation 4.7) is satisfied under very mild conditions. So, this provides a convenient way to give a useful functional form to σ_x, without knowing in advance what values of a and b might be appropriate. Simulations (Roberts and Rosenthal, 2009) indicate that this adaptive algorithm works well, at least in simple examples.

Another approach, sometimes called the regional adaptive Metropolis algorithm (RAMA), use a finite partition of the state space: $\mathcal{X} = \mathcal{X}_1 \dot{\cup} \ldots \dot{\cup} \mathcal{X}_m$. The proposal scaling is then given by $\sigma_x = e^{a_i}$ whenever $x \in \mathcal{X}_i$, with the acceptance probability (Equation 4.8) computed accordingly. Each of the values a_i is again adapted after each batch of iterations, by adding or subtracting $\delta(n)$ in an attempt to make the acceptance fraction of proposals from $\mathcal{X}_i$ close to 0.234. (As a special case, if there were no visits to $\mathcal{X}_i$ during the batch, then we always *add* $\delta(n)$ to a_i, to avoid the problem of a_i becoming so low that proposed moves to $\mathcal{X}_i$ are never accepted.) Once again, the algorithm will be valid under very mild conditions provided $\delta(n) \to 0$.

Recent work of Craiu et al. (2009) considers certain modifications of RAMA, in which multiple copies of the algorithm are run simultaneously in an effort to be sure to "learn" about *all* modes rather than getting stuck in a single mode. Their work also allows the proposal distribution to be a weighted mixture of the different $N(x, e^{2a_i})$, to allow for the possibility that the partition $\{\mathcal{X}_i\}$ was imperfectly chosen. It appears that such greater flexibility will allow for wider applicability of RAMA-type algorithms.

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

4.3.5 Limit Theorems

Many applications of MCMC make use of such Markov chain limit theorems as the weak law of large numbers (WLLN), strong law of large numbers (SLLN), and central limit theorem (CLT), in order to guarantee good asymptotic estimates and estimate standard errors (see, e.g. Tierney, 1994; Jones and Hobert, 2001; Hobert et al., 2002; Jones, 2004; Roberts and

Rosenthal, 2004). So, it is natural to ask if such limit theorems hold for adaptive MCMC as well.

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

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

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

4.3.6 Frequently Asked Questions

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

No. In particular, if the diminishing adaptation condition (Equation 4.6) does not hold, then there are simple counterexamples showing that adaptive MCMC can converge to the wrong answer, even though each individual Markov chain kernel would correctly converge to π.

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

Not really. As long as you satisfy diminishing adaptation (Equation 4.6), which is important but quite intuitive, then your algorithm will probably be asymptotically valid.

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

Yes, they have. Simulations on test problems involving hundreds of dimensions have been quite successful (Roberts and Rosenthal, 2009), and adaptive Metropolis-within-Gibbs has also been used on statistical genetics problems (Turro et al., 2007).

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

No. The ergodicity results presented herein do *not* require that the parameters $\{\Gamma_n\}$ converge at all, only that they satisfy (Equation 4.6) which still allows for the possibility of infinite total adaptation. However, many of the specific adaptive MCMC algorithms proposed are indeed designed to attempt to converge to specific values (e.g. to proposal scalings which give an asymptotic acceptance rate of 0.234).

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

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

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

Yes, it will (provided each individual kernel P_γ is ergodic), and this is a sensible method to try. However, it may be unclear how much adaptation should be done before you stop. For example, with adaptive Metropolis in 200 dimensions, it took well over a million iterations (Roberts and Rosenthal, 2009) before a truly good proposal covariance matrix was found—and it was not clear *a priori* that it would take nearly so long.

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

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

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

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

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

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

4.4 Conclusion

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

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

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

Overall, we feel that these results indicate the widespread applicability of both optimal scaling and adaptive MCMC algorithms to many different MCMC settings (Roberts and Rosenthal, 2009; Turro et al., 2007), including to complicated high-dimensional distributions. We hope that many MCMC users will be guided by optimal scaling results, and experiment with adaptive algorithms, in their future applications.

References

Andrieu, C. and Atchadé, Y. F. 2007. On the efficiency of adaptive MCMC algorithms. *Electronic Communications in Probability*, 12:336–349.

Andrieu, C. and Moulines, E. 2006. On the ergodicity properties of some adaptive Markov chain Monte Carlo algorithms. *Annals of Applied Probability*, 16:1462–1505.

Atchadé, Y. F. 2006. An adaptive version for the Metropolis adjusted Langevin algorithm with a truncated drift. *Methodology and Computing in Applied Probability*, 8:235–254.

Atchadé, Y. F. 2007. A cautionary tale on the efficiency of some adaptive Monte Carlo schemes. *Annals of Applied Probability*, to appear.

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

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

Bédard, M. 2007. Weak convergence of Metropolis algorithms for non-iid target distributions. *Annals of Applied Probability*, 17:1222–1244.

Bédard, M. 2008a. Efficient sampling using Metropolis algorithms: Applications of optimal scaling results. *Journal of Computational and Graphical Statistics*, 17:1–21.

Bédard, M. 2008b. Optimal acceptance rates for Metropolis algorithms: Moving beyond 0.234. *Stochastic Processes and their Applications*, 118(12):2198–2222.

Bédard, M. and Rosenthal, J. S. 2008. Optimal scaling of Metropolis algorithms: Heading towards general target distributions. *Canadian Journal of Statistics*, 36(4):483–503.

Cappé, O., Guillin, A., Marin, J. M., and Robert, C. P. 2004. Population Monte Carlo. *Journal of Computational and Graphical Statistics*, 13:907–930.

Craiu, R. V., Rosenthal, J. S., and Yang, C. 2009. Learn from thy neighbor: Parallel-chain adaptive MCMC. *Journal of the American Statistical Association*, 488:1454–1466.

Del Moral, P., Doucet, A., and Jasra, A. 2006. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society, Series B*, 68:411–436.

Gelman, A., Roberts, G. O., and Gilks, W. R. 1996. Efficient Metropolis jumping rules. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith (eds), *Bayesian Statistics 5: Proceedings of the Fifth Valencia International Meeting*, pp. 599–607. Oxford University Press, Oxford.

Giordani, P. and Kohn, R. 2006. Adaptive independent Metropolis-Hastings by fast estimation of mixtures of normals. *Journal of Computational and Graphical Statistics*, to appear.

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

Haario, H., Saksman, E., and Tamminen, J. 2005. Componentwise adaptation for high dimensional MCMC. *Computational Statistics*, 20:265–274.

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

Hobert, J. P., Jones, G. L., Presnell, B., and Rosenthal, J. S. 2002. On the applicability of regenerative simulation in Markov chain Monte Carlo. *Biometrika* 89:731–743.

Jones, G. L. 2004. On the Markov chain central limit theorem. *Probability Surveys*, 1:299–320.

Jones, G. L. and Hobert, J. P. 2001. Honest exploration of intractable probability distributions via Markov chain Monte Carlo. *Statistical Science*, 16(4):312–334.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092.

Mira, A. 2001. Ordering and improving the performance of Monte Carlo Markov chains. *Statistical Science*, 16:340–350.

Neal, P. J., Roberts, G. O., and Yuen, W. K. 2007. Optimal scaling of random walk Metropolis algorithms with discontinuous target densities. Preprint.

Pasarica, C. and Gelman, A. 2010. Adaptively scaling the Metropolis algorithm using expected squared jumped distance. *Statistica Sinica*, 20:343–364.

Pitt, M. K. and Shephard, N. 1999. Filtering via simulation: auxiliary particle filters. *Journal of the American Statistical Association*, 94:1403–1412.

Roberts, G. O., Gelman, A., and Gilks, W. R. 1997. Weak convergence and optimal scaling of random walk Metropolis algorithms. *Annals of Applied Probability*, 7:110–120.

Roberts, G. O. and Rosenthal, J. S. 1998. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society*, *Series B*, 60:255–268.

Roberts, G. O. and Rosenthal, J. S. 2001. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16:351–367.

Roberts, G. O. and Rosenthal, J. S. 2004. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20–71.

Roberts, G. O. and Rosenthal, J. S. 2007. Coupling and ergodicity of adaptive MCMC. *Journal of Applied Probability* 44:458–475.

Roberts, G. O. and Rosenthal, J. S. 2009. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics* 18(2):349–367.

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

Roberts, G. O. and Tweedie, R. L. 1999. Bounds on regeneration times and convergence rates for Markov chains. *Stochastic Processes and Their Applications*, 80: 211–229. Corrigendum (2001), 91:337–338.

Rosenthal, J. S. 1995. Minorization conditions and convergence rates for Markov chain Monte Carlo. *Journal of the American Statistical Association*, 90:558–566.

Rosenthal, J. S. 2002. Quantitative convergence rates of Markov chains: A simple account. *Electronic Communications in Probability*, 7(13):123–128.

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

Rosenthal, J. S. 2007. AMCMC: An R interface for adaptive MCMC. *Computational Statistics & Data Analysis*, 51:5467–5470. (Related software available at probability.ca/amcmc.)

Sherlock, C. 2006. Methodology for inference on the Markov modulated Poisson processes and theory for optimal scaling of the random walk Metropolis. Ph.D. dissertation, Lancaster University.

Tierney, L. 1994. Markov chains for exploring posterior distributions (with discussion). *Annals of Statistics*, 22:1701–1762.

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

Yang, C. 2008a. On the weak law of large numbers for unbounded functionals for adaptive MCMC. Preprint.

Yang, C. 2008b. Recurrent and ergodic properties of adaptive MCMC. Preprint.

5

MCMC Using Hamiltonian Dynamics

Radford M. Neal

5.1 Introduction

Markov chain Monte Carlo (MCMC) originated with the classic paper of Metropolis et al. (1953), where it was used to simulate the distribution of states for a system of idealized molecules. Not long after, another approach to molecular simulation was introduced (Alder and Wainwright, 1959), in which the motion of the molecules was deterministic, following Newton's laws of motion, which have an elegant formalization as *Hamiltonian dynamics*. For finding the properties of bulk materials, these approaches are asymptotically equivalent, since even in a deterministic simulation, each local region of the material experiences effectively random influences from distant regions. Despite the large overlap in their application areas, the MCMC and molecular dynamics approaches have continued to coexist in the following decades (see Frenkel and Smit, 1996).

In 1987, a landmark paper by Duane, Kennedy, Pendleton, and Roweth united the MCMC and molecular dynamics approaches. They called their method "hybrid Monte Carlo," which abbreviates to "HMC," but the phrase "Hamiltonian Monte Carlo," retaining the abbreviation, is more specific and descriptive, and I will use it here. Duane et al. applied HMC not to molecular simulation, but to lattice field theory simulations of quantum chromodynamics. Statistical applications of HMC began with my use of it for neural network models (Neal, 1996a). I also provided a statistically-oriented tutorial on HMC in a review of MCMC methods (Neal, 1993, Chapter 5). There have been other applications of HMC to statistical problems (e.g. Ishwaran, 1999; Schmidt, 2009) and statistically-oriented reviews (e.g. Liu, 2001, Chapter 9), but HMC still seems to be underappreciated by statisticians, and perhaps also by physicists outside the lattice field theory community.

This review begins by describing Hamiltonian dynamics. Despite terminology that may be unfamiliar outside physics, the features of Hamiltonian dynamics that are needed for HMC are elementary. The differential equations of Hamiltonian dynamics must be discretized for computer implementation. The "leapfrog" scheme that is typically used is quite simple.

Following this introduction to Hamiltonian dynamics, I describe how to use it to construct an MCMC method. The first step is to define a Hamiltonian function in terms of the probability distribution we wish to sample from. In addition to the variables we are interested in (the "position" variables), we must introduce auxiliary "momentum" variables, which typically have independent Gaussian distributions. The HMC method alternates simple updates for these momentum variables with Metropolis updates in which a new state is proposed by computing a trajectory according to Hamiltonian dynamics, implemented with the leapfrog method. A state proposed in this way can be distant from the

current state but nevertheless have a high probability of acceptance. This bypasses the slow exploration of the state space that occurs when Metropolis updates are done using a simple random-walk proposal distribution. (An alternative way of avoiding random walks is to use short trajectories but only partially replace the momentum variables between trajectories, so that successive trajectories tend to move in the same direction.)

After presenting the basic HMC method, I discuss practical issues of tuning the leapfrog stepsize and number of leapfrog steps, as well as theoretical results on the scaling of HMC with dimensionality. I then present a number of variations on HMC. The acceptance rate for HMC can be increased for many problems by looking at "windows" of states at the beginning and end of the trajectory. For many statistical problems, approximate computation of trajectories (e.g. using subsets of the data) may be beneficial. Tuning of HMC can be made easier using a "short-cut" in which trajectories computed with a bad choice of stepsize take little computation time. Finally, "tempering" methods may be useful when multiple isolated modes exist.

5.2 Hamiltonian Dynamics

Hamiltonian dynamics has a physical interpretation that can provide useful intuitions. In two dimensions, we can visualize the dynamics as that of a frictionless puck that slides over a surface of varying height. The state of this system consists of the *position* of the puck, given by a two-dimensional vector q, and the *momentum* of the puck (its mass times its velocity), given by a two-dimensional vector p. The *potential energy*, $U(q)$, of the puck is proportional to the height of the surface at its current position, and its *kinetic energy*, $K(p)$, is equal to $|p|^2/(2m)$, where m is the mass of the puck. On a level part of the surface, the puck moves at a constant velocity, equal to p/m. If it encounters a rising slope, the puck's momentum allows it to continue, with its kinetic energy decreasing and its potential energy increasing, until the kinetic energy (and hence p) is zero, at which point it will slide back down (with kinetic energy increasing and potential energy decreasing).

In nonphysical MCMC applications of Hamiltonian dynamics, the position will correspond to the variables of interest. The potential energy will be minus the log of the probability density for these variables. Momentum variables, one for each position variable, will be introduced artificially.

These interpretations may help motivate the exposition below, but if you find otherwise, the dynamics can also be understood as simply resulting from a certain set of differential equations.

5.2.1 Hamilton's Equations

Hamiltonian dynamics operates on a d-dimensional *position* vector, q, and a d-dimensional *momentum* vector, p, so that the full state space has $2d$ dimensions. The system is described by a function of q and p known as the *Hamiltonian*, $H(q, p)$.

5.2.1.1 Equations of Motion

The partial derivatives of the Hamiltonian determine how q and p change over time, t, according to Hamilton's equations:

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}, \tag{5.1}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}, \tag{5.2}$$

for $i = 1, \ldots, d$. For any time interval of duration s, these equations define a mapping, T_s, from the state at any time t to the state at time $t + s$. (Here, H, and hence T_s, are assumed to not depend on t.)

Alternatively, we can combine the vectors q and p into the vector $z = (q, p)$ with $2d$ dimensions, and write Hamilton's equations as

$$\frac{dz}{dt} = J\,\nabla H(z),$$

where ∇H is the gradient of H (i.e. $[\nabla H]_k = \partial H/\partial z_k$), and

$$J = \begin{bmatrix} 0_{d\times d} & I_{d\times d} \\ -I_{d\times d} & 0_{d\times d} \end{bmatrix} \tag{5.3}$$

is a $2d \times 2d$ matrix whose quadrants are defined above in terms of identity and zero matrices.

5.2.1.2 Potential and Kinetic Energy

For HMC we usually use Hamiltonian functions that can be written as

$$H(q, p) = U(q) + K(p). \tag{5.4}$$

Here $U(q)$ is called the *potential energy*, and will be defined to be minus the log probability density of the distribution for q that we wish to sample, plus any constant that is convenient. $K(p)$ is called the *kinetic energy*, and is usually defined as

$$K(p) = p^T M^{-1} p/2. \tag{5.5}$$

Here M is a symmetric, positive-definite "mass matrix," which is typically diagonal, and is often a scalar multiple of the identity matrix. This form for $K(p)$ corresponds to minus the log probability density (plus a constant) of the zero-mean Gaussian distribution with covariance matrix M.

With these forms for H and K, Hamilton's equations 5.1 and 5.2 can be written as follows, for $i = 1, \ldots, d$:

$$\frac{dq_i}{dt} = [M^{-1}p]_i, \tag{5.6}$$

$$\frac{dp_i}{dt} = -\frac{\partial U}{\partial q_i}. \tag{5.7}$$

5.2.1.3 A One-Dimensional Example

Consider a simple example in one dimension (for which q and p are scalars and will be written without subscripts), in which the Hamiltonian is defined as follows:

$$H(q,p) = U(q) + K(p), \qquad U(q) = \frac{q^2}{2}, \quad K(p) = \frac{p^2}{2}. \tag{5.8}$$

As we will see later in Section 5.3.1, this corresponds to a Gaussian distribution for q with mean zero and variance one. The dynamics resulting from this Hamiltonian (following Equations 5.6 and 5.7) is

$$\frac{dq}{dt} = p, \quad \frac{dp}{dt} = -q.$$

Solutions have the following form, for some constants r and a:

$$q(t) = r\cos(a+t), \quad p(t) = -r\sin(a+t). \tag{5.9}$$

Hence, the mapping T_s is a rotation by s radians clockwise around the origin in the (q,p) plane. In higher dimensions, Hamiltonian dynamics generally does not have such a simple periodic form, but this example does illustrate some important properties that we will look at next.

5.2.2 Properties of Hamiltonian Dynamics

Several properties of Hamiltonian dynamics are crucial to its use in constructing MCMC updates.

5.2.2.1 Reversibility

First, Hamiltonian dynamics is *reversible*—the mapping T_s from the state at time t, $(q(t), p(t))$, to the state at time $t+s$, $(q(t+s), p(t+s))$, is one-to-one, and hence has an inverse, T_{-s}. This inverse mapping is obtained by simply negating the time derivatives in Equations 5.1 and 5.2. When the Hamiltonian has the form in Equation 5.4, and $K(p) = K(-p)$, as in the quadratic form for the kinetic energy of Equation 5.5, the inverse mapping can also be obtained by negating p, applying T_s, and then negating p again.

In the simple one-dimensional example of Equation 5.8, T_{-s} is just a counterclockwise rotation by s radians, undoing the clockwise rotation of T_s.

The reversibility of Hamiltonian dynamics is important for showing that MCMC updates that use the dynamics leave the desired distribution invariant, since this is most easily proved by showing reversibility of the Markov chain transitions, which requires reversibility of the dynamics used to propose a state.

5.2.2.2 Conservation of the Hamiltonian

A second property of the dynamics is that it *keeps the Hamiltonian invariant* (i.e. conserved). This is easily seen from Equations 5.1 and 5.2 as follows:

$$\frac{dH}{dt} = \sum_{i=1}^{d}\left[\frac{dq_i}{dt}\frac{\partial H}{\partial q_i} + \frac{dp_i}{dt}\frac{\partial H}{\partial p_i}\right] = \sum_{i=1}^{d}\left[\frac{\partial H}{\partial p_i}\frac{\partial H}{\partial q_i} - \frac{\partial H}{\partial q_i}\frac{\partial H}{\partial p_i}\right] = 0. \tag{5.10}$$

With the Hamiltonian of Equation 5.8, the value of the Hamiltonian is half the squared distance from the origin, and the solutions (Equation 5.9) stay at a constant distance from the origin, keeping H constant.

For Metropolis updates using a proposal found by Hamiltonian dynamics, which form part of the HMC method, the acceptance probability is one if H is kept invariant. We will see later, however, that in practice we can only make H approximately invariant, and hence we will not quite be able to achieve this.

5.2.2.3 Volume Preservation

A third fundamental property of Hamiltonian dynamics is that it *preserves volume* in (q, p) space (a result known as Liouville's theorem). If we apply the mapping T_s to the points in some region R of (q, p) space, with volume V, the image of R under T_s will also have volume V.

With the Hamiltonian of Equation 5.8, the solutions (Equation 5.9) are rotations, which obviously do not change the volume. Such rotations also do not change the shape of a region, but this is not so in general—Hamiltonian dynamics might stretch a region in one direction, as long as the region is squashed in some other direction so as to preserve volume.

The significance of volume preservation for MCMC is that we need not account for any change in volume in the acceptance probability for Metropolis updates. If we proposed new states using some arbitrary, non-Hamiltonian, dynamics, we would need to compute the determinant of the Jacobian matrix for the mapping the dynamics defines, which might well be infeasible.

The preservation of volume by Hamiltonian dynamics can be proved in several ways. One is to note that the divergence of the vector field defined by Equations 5.1 and 5.2 is zero, which can be seen as follows:

$$\sum_{i=1}^{d} \left[\frac{\partial}{\partial q_i} \frac{dq_i}{dt} + \frac{\partial}{\partial p_i} \frac{dp_i}{dt} \right] = \sum_{i=1}^{d} \left[\frac{\partial}{\partial q_i} \frac{\partial H}{\partial p_i} - \frac{\partial}{\partial p_i} \frac{\partial H}{\partial q_i} \right] = \sum_{i=1}^{d} \left[\frac{\partial^2 H}{\partial q_i \partial p_i} - \frac{\partial^2 H}{\partial p_i \partial q_i} \right] = 0.$$

A vector field with zero divergence can be shown to preserve volume (Arnold, 1989).

Here, I will show informally that Hamiltonian dynamics preserves volume more directly, without presuming this property of the divergence. I will, however, take as given that volume preservation is equivalent to the determinant of the Jacobian matrix of T_s having absolute value one, which is related to the well-known role of this determinant in regard to the effect of transformations on definite integrals and on probability density functions.

The $2d \times 2d$ Jacobian matrix of T_s, seen as a mapping of $z = (q, p)$, will be written as B_s. In general, B_s will depend on the values of q and p before the mapping. When B_s is diagonal, it is easy to see that the absolute values of its diagonal elements are the factors by which T_s stretches or compresses a region in each dimension, so that the product of these factors, which is equal to the absolute value of $\det(B_s)$, is the factor by which the volume of the region changes. I will not prove the general result here, but note that if we were to (say) rotate the coordinate system used, B_s would no longer be diagonal, but the determinant of B_s is invariant to such transformations, and so would still give the factor by which the volume changes.

Let us first consider volume preservation for Hamiltonian dynamics in one dimension (i.e. with $d = 1$), for which we can drop the subscripts on p and q. We can approximate T_δ

for δ near zero as follows:

$$T_\delta(q,p) = \begin{bmatrix} q \\ p \end{bmatrix} + \delta \begin{bmatrix} dq/dt \\ dp/dt \end{bmatrix} + \text{terms of order } \delta^2 \text{ or higher.}$$

Taking the time derivatives from Equations 5.1 and 5.2, the Jacobian matrix can be written as

$$B_\delta = \begin{bmatrix} 1 + \delta \dfrac{\partial^2 H}{\partial q \partial p} & \delta \dfrac{\partial^2 H}{\partial p^2} \\ -\delta \dfrac{\partial^2 H}{\partial q^2} & 1 - \delta \dfrac{\partial^2 H}{\partial p \partial q} \end{bmatrix} + \text{terms of order } \delta^2 \text{ or higher.} \tag{5.11}$$

We can then write the determinant of this matrix as

$$\begin{aligned} \det(B_\delta) &= 1 + \delta \frac{\partial^2 H}{\partial q \partial p} - \delta \frac{\partial^2 H}{\partial p \partial q} + \text{terms of order } \delta^2 \text{ or higher} \\ &= 1 + \text{terms of order } \delta^2 \text{ or higher.} \end{aligned}$$

Since $\log(1+x) \approx x$ for x near zero, $\log\det(B_\delta)$ is zero, except perhaps for terms of order δ^2 or higher (though we will see later that it is exactly zero). Now consider $\log\det(B_s)$ for some time interval s that is not close to zero. Setting $\delta = s/n$, for some integer n, we can write T_s as the composition of T_δ applied n times (from n points along the trajectory), so $\det(B_s)$ is the n-fold product of $\det(B_\delta)$ evaluated at these points. We then find that

$$\begin{aligned} \log\det(B_s) &= \sum_{i=1}^{n} \log\det(B_\delta) \\ &= \sum_{i=1}^{n} \Big\{ \text{terms of order } 1/n^2 \text{ or smaller} \Big\} \\ &= \text{terms of order } 1/n \text{ or smaller.} \end{aligned} \tag{5.12}$$

Note that the value of B_δ in the sum in Equation 5.12 might perhaps vary with i, since the values of q and p vary along the trajectory that produces T_s. However, assuming that trajectories are not singular, the variation in B_δ must be bounded along any particular trajectory. Taking the limit as $n \to \infty$, we conclude that $\log\det(B_s) = 0$, so $\det(B_s) = 1$, and hence T_s preserves volume.

When $d > 1$, the same argument applies. The Jacobian matrix will now have the following form (compare Equation 5.11), where each entry shown below is a $d \times d$ submatrix, with rows indexed by i and columns by j:

$$B_\delta = \begin{bmatrix} I + \delta \left[\dfrac{\partial^2 H}{\partial q_j \partial p_i} \right] & \delta \left[\dfrac{\partial^2 H}{\partial p_j \partial p_i} \right] \\ -\delta \left[\dfrac{\partial^2 H}{\partial q_j \partial q_i} \right] & I - \delta \left[\dfrac{\partial^2 H}{\partial p_j \partial q_i} \right] \end{bmatrix} + \text{terms of order } \delta^2 \text{ or higher.}$$

As for $d = 1$, the determinant of this matrix will be one plus terms of order δ^2 or higher, since all the terms of order δ cancel. The remainder of the argument above then applies without change.

5.2.2.4 Symplecticness

Volume preservation is also a consequence of Hamiltonian dynamics being *symplectic*. Letting $z = (q, p)$, and defining J as in Equation 5.3, the symplecticness condition is that the Jacobian matrix, B_s, of the mapping T_s satisfies

$$B_s^T J^{-1} B_s = J^{-1}.$$

This implies volume conservation, since $\det(B_s^T)\det(J^{-1})\det(B_s) = \det(J^{-1})$ implies that $\det(B_s)^2$ is one. When $d > 1$, the symplecticness condition is stronger than volume preservation. Hamiltonian dynamics and the symplecticness condition can be generalized to where J is any matrix for which $J^T = -J$ and $\det(J) \neq 0$.

Crucially, reversibility, preservation of volume, and symplecticness can be maintained exactly even when, as is necessary in practice, Hamiltonian dynamics is approximated, as we will see next.

5.2.3 Discretizing Hamilton's Equations—The Leapfrog Method

For computer implementation, Hamilton's equations must be approximated by discretizing time, using some small stepsize, ε. Starting with the state at time zero, we iteratively compute (approximately) the state at times $\varepsilon, 2\varepsilon, 3\varepsilon$, etc.

In discussing how to do this, I will assume that the Hamiltonian has the form $H(q, p) = U(q) + K(p)$, as in Equation 5.4. Although the methods below can be applied with any form for the kinetic energy, I assume for simplicity that $K(p) = p^T M^{-1} p/2$, as in Equation 5.5, and furthermore that M is diagonal, with diagonal elements $m_1, \ldots, m_d$, so that

$$K(p) = \sum_{i=1}^{d} \frac{p_i^2}{2m_i}. \tag{5.13}$$

5.2.3.1 Euler's Method

Perhaps the best-known way to approximate the solution to a system of differential equations is Euler's method. For Hamilton's equations, this method performs the following steps, for each component of position and momentum, indexed by $i = 1, \ldots, d$:

$$p_i(t + \varepsilon) = p_i(t) + \varepsilon \frac{dp_i}{dt}(t) = p_i(t) - \varepsilon \frac{\partial U}{\partial q_i}(q(t)), \tag{5.14}$$

$$q_i(t + \varepsilon) = q_i(t) + \varepsilon \frac{dq_i}{dt}(t) = q_i(t) + \varepsilon \frac{p_i(t)}{m_i}. \tag{5.15}$$

The time derivatives in Equations 5.14 and 5.15 are from the form of Hamilton's equations given by Equations 5.6 and 5.7. If we start at $t = 0$ with given values for $q_i(0)$ and $p_i(0)$, we can iterate the steps above to get a trajectory of position and momentum values

at times $\varepsilon, 2\varepsilon, 3\varepsilon, \ldots$, and hence find (approximate) values for $q(\tau)$ and $p(\tau)$ after τ/ε steps (assuming τ/ε is an integer).

Figure 5.1a shows the result of using Euler's method to approximate the dynamics defined by the Hamiltonian of Equation 5.8, starting from $q(0) = 0$ and $p(0) = 1$, and using a stepsize of $\varepsilon = 0.3$ for 20 steps (i.e. to $\tau = 0.3 \times 20 = 6$). The results are not good—Euler's method produces a trajectory that diverges to infinity, but the true trajectory is a circle. Using a smaller value of ε, and correspondingly more steps, produces a more accurate result at $\tau = 6$, but although the divergence to infinity is slower, it is not eliminated.

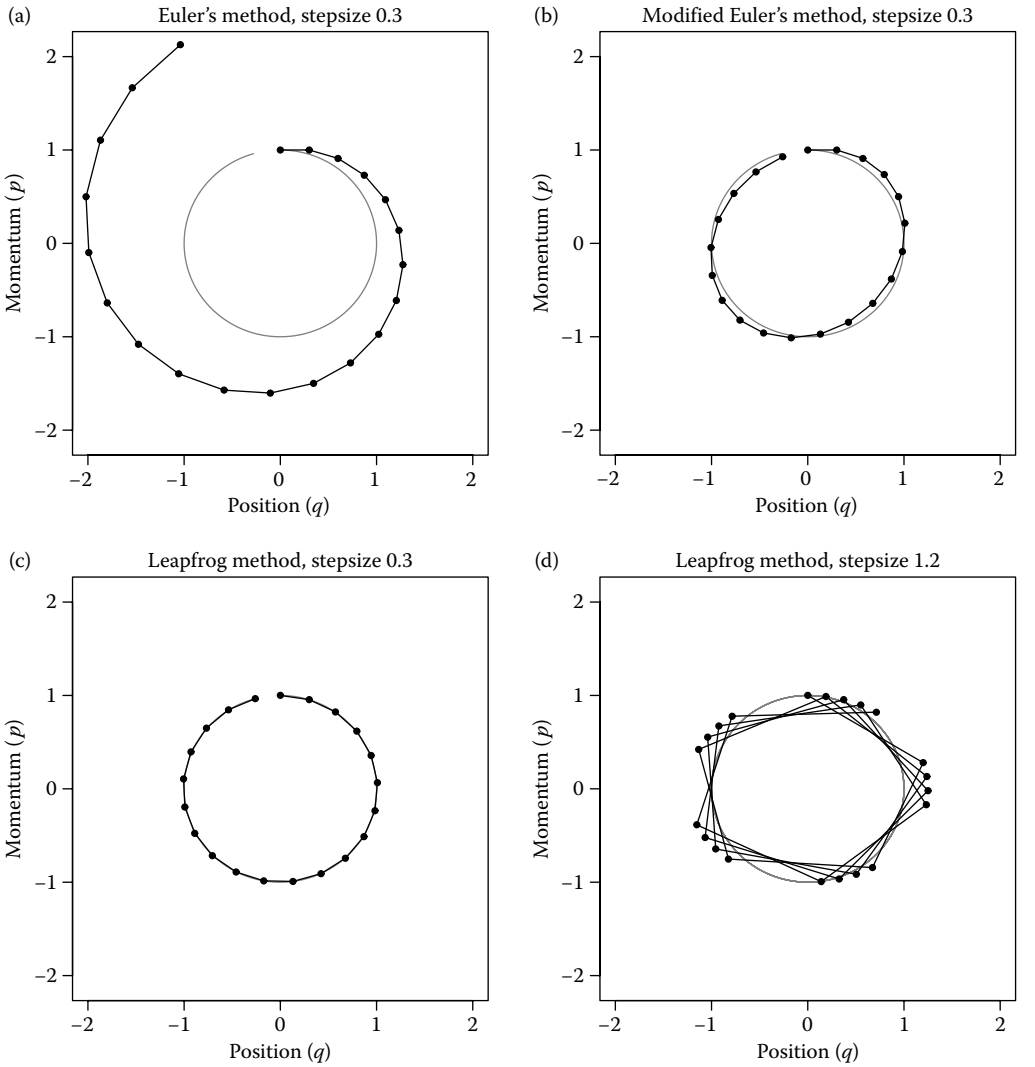

FIGURE 5.1
Results using three methods for approximating Hamiltonian dynamics, when $H(q, p) = q^2/2 + p^2/2$. The initial state was $q = 0, p = 1$. The stepsize was $\varepsilon = 0.3$ for (a), (b), and (c), and $\varepsilon = 1.2$ for (d). Twenty steps of the simulated trajectory are shown for each method, along with the true trajectory (in gray).

5.2.3.2 A Modification of Euler's Method

Much better results can be obtained by slightly modifying Euler's method, as follows:

$$p_i(t+\varepsilon) = p_i(t) - \varepsilon \frac{\partial U}{\partial q_i}(q(t)), \tag{5.16}$$

$$q_i(t+\varepsilon) = q_i(t) + \varepsilon \frac{p_i(t+\varepsilon)}{m_i}. \tag{5.17}$$

We simply use the *new* value for the momentum variables, p_i, when computing the new value for the position variables, q_i. A method with similar performance can be obtained by instead updating the q_i first and using their new values to update the p_i.

Figure 5.1b shows the results using this modification of Euler's method with $\varepsilon = 0.3$. Though not perfect, the trajectory it produces is much closer to the true trajectory than that obtained using Euler's method, with no tendency to diverge to infinity. This better performance is related to the modified method's exact preservation of volume, which helps avoid divergence to infinity or spiraling into the origin, since these would typically involve the volume expanding to infinity or contracting to zero.

To see that this modification of Euler's method preserves volume exactly despite the finite discretization of time, note that both the transformation from $(q(t), p(t))$ to $(q(t), p(t+\varepsilon))$ via Equation 5.16 and the transformation from $(q(t), p(t+\varepsilon))$ to $(q(t+\varepsilon), p(t+\varepsilon))$ via Equation 5.17 are "shear" transformations, in which only some of the variables change (either the p_i or the q_i), by amounts that depend only on the variables that do not change. Any shear transformation will preserve volume, since its Jacobian matrix will have determinant one (as the only nonzero term in the determinant will be the product of diagonal elements, which will all be one).

5.2.3.3 The Leapfrog Method

Even better results can be obtained with the *leapfrog* method, which works as follows:

$$p_i\,(t+\varepsilon/2) = p_i(t) - (\varepsilon/2)\ \frac{\partial U}{\partial q_i}(q(t)), \tag{5.18}$$

$$q_i(t+\varepsilon) = q_i(t) + \varepsilon \frac{p_i(t+\varepsilon/2)}{m_i}, \tag{5.19}$$

$$p_i(t+\varepsilon) = p_i\,(t+\varepsilon/2) - (\varepsilon/2)\ \frac{\partial U}{\partial q_i}(q(t+\varepsilon)). \tag{5.20}$$

We start with a half step for the momentum variables, then do a full step for the position variables, using the new values of the momentum variables, and finally do another half step for the momentum variables, using the new values for the position variables. An analogous scheme can be used with any kinetic energy function, with $\partial K/\partial p_i$ replacing p_i/m_i above.

When we apply Equations 5.18 through 5.20 a second time to go from time $t+\varepsilon$ to $t+2\varepsilon$, we can combine the last half step of the first update, from $p_i(t+\varepsilon/2)$ to $p_i(t+\varepsilon)$, with the first half step of the second update, from $p_i(t+\varepsilon)$ to $p_i(t+\varepsilon+\varepsilon/2)$. The leapfrog method then looks very similar to the modification of Euler's method in Equations 5.17 and 5.16, except that leapfrog performs half steps for momentum at the very beginning and very end of the trajectory, and the time labels of the momentum values computed are shifted by $\varepsilon/2$.

The leapfrog method preserves volume exactly, since Equations 5.18 through 5.20 are shear transformations. Due to its symmetry, it is also reversible by simply negating p, applying the same number of steps again, and then negating p again.

Figure 5.1c shows the results using the leapfrog method with a stepsize of $\varepsilon = 0.3$, which are indistinguishable from the true trajectory, at the scale of this plot. In Figure 5.1d, the results of using the leapfrog method with $\varepsilon = 1.2$ are shown (still with 20 steps, so almost four cycles are seen, rather than almost one). With this larger stepsize, the approximation error is clearly visible, but the trajectory still remains stable (and will stay stable indefinitely). Only when the stepsize approaches $\varepsilon = 2$ do the trajectories become unstable.

5.2.3.4 Local and Global Error of Discretization Methods

I will briefly discuss how the error from discretizing the dynamics behaves in the limit as the stepsize, ε, goes to zero; Leimkuhler and Reich (2004) provide a much more detailed discussion. For useful methods, the error goes to zero as ε goes to zero, so that any upper limit on the error will apply (apart from a usually unknown constant factor) to any differentiable function of state—for example, if the error for (q, p) is no more than order ε^2, the error for $H(q, p)$ will also be no more than order ε^2.

The *local error* is the error after one step, that moves from time t to time $t + \varepsilon$. The *global error* is the error after simulating for some fixed time interval, s, which will require s/ε steps. If the local error is order ε^p, the global error will be order ε^{p-1}—the local errors of order ε^p accumulate over the s/ε steps to give an error of order ε^{p-1}. If we instead fix ε and consider increasing the time, s, for which the trajectory is simulated, the error can in general increase exponentially with s. Interestingly, however, this is often not what happens when simulating Hamiltonian dynamics with a symplectic method, as can be seen in Figure 5.1.

The Euler method and its modification above have order ε^2 local error and order ε global error. The leapfrog method has order ε^3 local error and order ε^2 global error. As shown by Leimkuhler and Reich (2004, Section 4.3.3), this difference is a consequence of leapfrog being reversible, since any reversible method must have global error that is of even order in ε.

5.3 MCMC from Hamiltonian Dynamics

Using Hamiltonian dynamics to sample from a distribution requires translating the density function for this distribution to a potential energy function and introducing "momentum" variables to go with the original variables of interest (now seen as "position" variables). We can then simulate a Markov chain in which each iteration resamples the momentum and then does a Metropolis update with a proposal found using Hamiltonian dynamics.

5.3.1 Probability and the Hamiltonian: Canonical Distributions

The distribution we wish to sample can be related to a potential energy function via the concept of a *canonical distribution* from statistical mechanics. Given some energy function, $E(x)$, for the state, x, of some physical system, the canonical distribution over states has probability or probability density function

$$P(x) = \frac{1}{Z} \exp\left(\frac{-E(x)}{T}\right). \tag{5.21}$$

Here, T is the temperature of the system,* and Z is the normalizing constant needed for this function to sum or integrate to one. Viewing this the opposite way, if we are interested in some distribution with density function $P(x)$, we can obtain it as a canonical distribution with $T = 1$ by setting $E(x) = -\log P(x) - \log Z$, where Z is any convenient positive constant.

The Hamiltonian is an energy function for the joint state of "position," q, and "momentum," p, and so defines a joint distribution for them as follows:

$$P(q,p) = \frac{1}{Z} \exp\left(\frac{-H(q,p)}{T}\right).$$

Note that the invariance of H under Hamiltonian dynamics means that a Hamiltonian trajectory will (if simulated exactly) move within a hypersurface of constant probability density.

If $H(q,p) = U(q) + K(p)$, the joint density is

$$P(q,p) = \frac{1}{Z} \exp\left(\frac{-U(q)}{T}\right) \exp\left(\frac{-K(p)}{T}\right), \tag{5.22}$$

and we see that q and p are independent, and each have canonical distributions, with energy functions $U(q)$ and $K(p)$. We will use q to represent the variables of interest, and introduce p just to allow Hamiltonian dynamics to operate.

In Bayesian statistics, the posterior distribution for the model parameters is the usual focus of interest, and hence these parameters will take the role of the position, q. We can express the posterior distribution as a canonical distribution (with $T = 1$) using a potential energy function defined as

$$U(q) = -\log\left[\pi(q)L(q \mid D)\right],$$

where $\pi(q)$ is the prior density, and $L(q|D)$ is the likelihood function given data D.

5.3.2 The Hamiltonian Monte Carlo Algorithm

We now have the background needed to present the Hamiltonian Monte Carlo algorithm. HMC can be used to sample only from continuous distributions on $\mathbb{R}^d$ for which the density function can be evaluated (perhaps up to an unknown normalizing constant). For the moment, I will also assume that the density is nonzero everywhere (but this is relaxed in Section 5.5.1). We must also be able to compute the partial derivatives of the log of the density function. These derivatives must therefore exist, except perhaps on a set of points with probability zero, for which some arbitrary value could be returned.

HMC samples from the canonical distribution for q and p defined by Equation 5.22, in which q has the distribution of interest, as specified using the potential energy function $U(q)$. We can choose the distribution of the momentum variables, p, which are independent of q, as we wish, specifying the distribution via the kinetic energy function, $K(p)$. Current practice with HMC is to use a quadratic kinetic energy, as in Equation 5.5, which leads p to have a zero-mean multivariate Gaussian distribution. Most often, the components of

* Note to physicists: I assume here that temperature is measured in units that make Boltzmann's constant unity.

p are specified to be independent, with component i having variance m_i. The kinetic energy function producing this distribution (setting $T = 1$) is

$$K(p) = \sum_{i=1}^{d} \frac{p_i^2}{2m_i}. \tag{5.23}$$

We will see in Section 5.4 how the choice for the m_i affects performance.

5.3.2.1 The Two Steps of the HMC Algorithm

Each iteration of the HMC algorithm has two steps. The first changes only the momentum; the second may change both position and momentum. Both steps leave the canonical joint distribution of (q, p) invariant, and hence their combination also leaves this distribution invariant.

In the first step, new values for the momentum variables are randomly drawn from their Gaussian distribution, independently of the current values of the position variables. For the kinetic energy of Equation 5.23, the d momentum variables are independent, with p_i having mean zero and variance m_i. Since q is not changed, and p is drawn from its correct conditional distribution given q (the same as its marginal distribution, due to independence), this step obviously leaves the canonical joint distribution invariant.

In the second step, a Metropolis update is performed, using Hamiltonian dynamics to propose a new state. Starting with the current state, (q, p), Hamiltonian dynamics is simulated for L steps using the leapfrog method (or some other reversible method that preserves volume), with a stepsize of ε. Here, L and ε are parameters of the algorithm, which need to be tuned to obtain good performance (as discussed below in Section 5.4.2). The momentum variables at the end of this L-step trajectory are then negated, giving a proposed state (q^*, p^*). This proposed state is accepted as the next state of the Markov chain with probability

$$\min\left[1, \exp(-H(q^*, p^*) + H(q, p))\right] = \min\left[1, \exp(-U(q^*) + U(q) - K(p^*) + K(p))\right].$$

If the proposed state is not accepted (i.e. it is rejected), the next state is the same as the current state (and is counted again when estimating the expectation of some function of state by its average over states of the Markov chain). The negation of the momentum variables at the end of the trajectory makes the Metropolis proposal symmetrical, as needed for the acceptance probability above to be valid. This negation need not be done in practice, since $K(p) = K(-p)$, and the momentum will be replaced before it is used again, in the first step of the next iteration. (This assumes that these HMC updates are the only ones performed.)

If we look at HMC as sampling from the joint distribution of q and p, the Metropolis step using a proposal found by Hamiltonian dynamics leaves the probability density for (q, p) unchanged or almost unchanged. Movement to (q, p) points with a different probability density is accomplished only by the first step in an HMC iteration, in which p is replaced by a new value. Fortunately, this replacement of p can change the probability density for (q, p) by a large amount, so movement to points with a different probability density is not a problem (at least not for this reason). Looked at in terms of q only, Hamiltonian dynamics for (q, p) can produce a value for q with a much different probability density (equivalently, a much different potential energy, $U(q)$). However, the resampling of the momentum variables is still crucial to obtaining the proper distribution for q. Without resampling, $H(q, p) = U(q) + K(p)$ will be (nearly) constant, and since $K(p)$ and $U(q)$ are

```
HMC = function (U, grad_U, epsilon, L, current_q)
{
  q = current_q
  p = rnorm(length(q),0,1)  # independent standard normal variates
  current_p = p

  # Make a half step for momentum at the beginning
  p = p - epsilon * grad_U(q) / 2

  # Alternate full steps for position and momentum

  for (i in 1:L)
  {
    # Make a full step for the position
    q = q + epsilon * p
    # Make a full step for the momentum, except at end of trajectory
    if (i!=L) p = p - epsilon * grad_U(q)
  }

  # Make a half step for momentum at the end.
  p = p - epsilon * grad_U(q) / 2
  # Negate momentum at end of trajectory to make the proposal symmetric
  p = -p

  # Evaluate potential and kinetic energies at start and end of trajectory

  current_U = U(current_q)
  current_K = sum(current_p^2) / 2
  proposed_U = U(q)
  proposed_K = sum(p^2) / 2

  # Accept or reject the state at end of trajectory, returning either
  # the position at the end of the trajectory or the initial position

  if (runif(1) < exp(current_U-proposed_U+current_K-proposed_K))
  {
    return (q)  # accept
  }
  else
  {
    return (current_q)  # reject
  }
}
```

FIGURE 5.2
The Hamiltonian Monte Carlo algorithm.

nonnegative, $U(q)$ could never exceed the initial value of $H(q,p)$ if no resampling for p were done.

A function that implements a single iteration of the HMC algorithm, written in the R language,* is shown in Figure 5.2. Its first two arguments are functions: `U`, which returns

* R is available for free from www.r-project.org

the potential energy given a value for q, and `grad_U`, which returns the vector of partial derivatives of U given q. Other arguments are the stepsize, `epsilon`, for leapfrog steps; the number of leapfrog steps in the trajectory, `L`; and the current position, `current_q`, that the trajectory starts from. Momentum variables are sampled within this function, and discarded at the end, with only the next position being returned. The kinetic energy is assumed to have the simplest form, $K(p) = \sum p_i^2/2$ (i.e. all m_i are one). In this program, all components of p and of q are updated simultaneously, using vector operations. This simple implementation of HMC is available from my web page,* along with other R programs with extra features helpful for practical use, and that illustrate some of the variants of HMC in Section 5.5.

5.3.2.2 Proof That HMC Leaves the Canonical Distribution Invariant

The Metropolis update above is reversible with respect to the canonical distribution for q and p (with $T = 1$), a condition also known as "detailed balance," and which can be phrased informally as follows. Suppose that we partition the (q, p) space into regions A_k, each with the same small volume V. Let the image of A_k with respect to the operation of L leapfrog steps, plus a negation of the momentum, be B_k. Due to the reversibility of the leapfrog steps, the B_k will also partition the space, and since the leapfrog steps preserve volume (as does negation), each B_k will also have volume V. Detailed balance holds if, for all i and j,

$$P(A_i)T(B_j \mid A_i) = P(B_j)T(A_i \mid B_j), \tag{5.24}$$

where P is probability under the canonical distribution, and $T(X|Y)$ is the conditional probability of proposing and then accepting a move to region X if the current state is in region Y. Clearly, when $i \neq j$, $T(A_i \mid B_j) = T(B_j \mid A_i) = 0$ and so Equation 5.24 will be satisfied. Since the Hamiltonian is continuous almost everywhere, in the limit as the regions A_k and B_k become smaller, the Hamiltonian becomes effectively constant within each region, with value H_X in region X, and hence the canonical probability density and the transition probabilities become effectively constant within each region as well. We can now rewrite Equation 5.24 for $i = j$ (say, both equal to k) as

$$\frac{V}{Z}\exp(-H_{A_k})\min\left[1, \exp(-H_{B_k}+H_{A_k})\right] = \frac{V}{Z}\exp(-H_{B_k})\min\left[1, \exp(-H_{A_k}+H_{B_k})\right],$$

which is easily seen to be true.

Detailed balance implies that this Metropolis update leaves the canonical distribution for q and p invariant. This can be seen as follows. Let $R(X)$ be the probability that the Metropolis update for a state in the small region X leads to rejection of the proposed state. Suppose that the current state is distributed according to the canonical distribution. The probability that the next state is in a small region B_k is the sum of the probability that the current state is in B_k and the update leads to rejection, and the probability that the current state is in some region from which a move to B_k is proposed and accepted. The probability of the next state

* www.cs.utoronto.ca/∼radford

being in B_k can therefore be written as

$$\begin{aligned} P(B_k)R(B_k) + \sum_i P(A_i)T(B_k|A_i) &= P(B_k)R(B_k) + \sum_i P(B_k)T(A_i|B_k) \\ &= P(B_k)R(B_k) + P(B_k)\sum_i T(A_i|B_k) \\ &= P(B_k)R(B_k) + P(B_k)(1 - R(B_k)) \\ &= P(B_k). \end{aligned}$$

The Metropolis update within HMC therefore leaves the canonical distribution invariant.

Since both the sampling of momentum variables and the Metropolis update with a proposal found by Hamiltonian dynamics leave the canonical distribution invariant, the HMC algorithm as a whole does as well.

5.3.2.3 Ergodicity of HMC

Typically, the HMC algorithm will also be "ergodic"—it will not be trapped in some subset of the state space, and hence will asymptotically converge to its (unique) invariant distribution. In an HMC iteration, any value can be sampled for the momentum variables, which can typically then affect the position variables in arbitrary ways. However, ergodicity can fail if the L leapfrog steps in a trajectory produce an exact periodicity for some function of state. For example, with the simple Hamiltonian of Equation 5.8, the exact solutions (given by Equation 5.9) are periodic with period 2π. Approximate trajectories found with L leapfrog steps with stepsize ε may return to the same position coordinate when $L\varepsilon$ is approximately 2π. HMC with such values for L and ε will not be ergodic. For nearby values of L and ε, HMC may be theoretically ergodic, but take a very long time to move about the full state space.

This potential problem of nonergodicity can be solved by randomly choosing ε or L (or both) from some fairly small interval (Mackenzie, 1989). Doing this routinely may be advisable. Although in real problems interactions between variables typically prevent any exact periodicities from occurring, near periodicities might still slow HMC considerably.

5.3.3 Illustrations of HMC and Its Benefits

I will now illustrate some practical issues with HMC, and demonstrate its potential to sample much more efficiently than simple methods such as random-walk Metropolis. I use simple Gaussian distributions for these demonstrations, so that the results can be compared with known values, but of course HMC is typically used for more complex distributions.

5.3.3.1 Trajectories for a Two-Dimensional Problem

Consider sampling from a distribution for two variables that is bivariate Gaussian, with means of zero, standard deviations of one, and correlation 0.95. We regard these as "position" variables, and introduce two corresponding "momentum" variables, defined to have a Gaussian distribution with means of zero, standard deviations of one, and zero correlation. We then define the Hamiltonian as

$$H(q,p) = q^T \Sigma^{-1} q/2 + p^T p/2, \quad \text{with } \Sigma = \begin{bmatrix} 1 & 0.95 \\ 0.95 & 1 \end{bmatrix}.$$

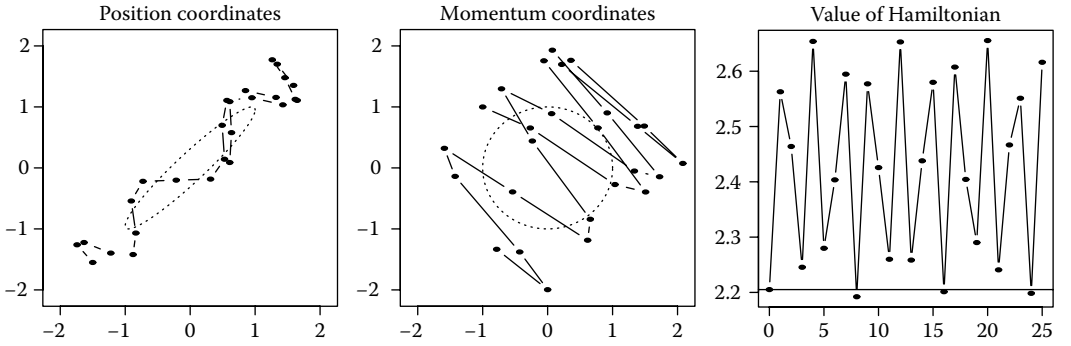

FIGURE 5.3
A trajectory for a two-dimensional Gaussian distribution, simulated using 25 leapfrog steps with a stepsize of 0.25. The ellipses plotted are one standard deviation from the means. The initial state had $q = [-1.50, -1.55]^T$ and $p = [-1, 1]^T$.

Figure 5.3 shows a trajectory based on this Hamiltonian, such as might be used to propose a new state in the HMC method, computed using $L = 25$ leapfrog steps, with a stepsize of $\varepsilon = 0.25$. Since the full state space is four-dimensional, Figure 5.3 shows the two position coordinates and the two momentum coordinates in separate plots, while the third plot shows the value of the Hamiltonian after each leapfrog step.

Notice that this trajectory does not resemble a random walk. Instead, starting from the lower left-hand corner, the position variables systematically move upward and to the right, until they reach the upper right-hand corner, at which point the direction of motion is reversed. The consistency of this motion results from the role of the momentum variables. The projection of p in the diagonal direction will change only slowly, since the gradient in that direction is small, and hence the direction of diagonal motion stays the same for many leapfrog steps. While this large-scale diagonal motion is happening, smaller-scale oscillations occur, moving back and forth across the "valley" created by the high correlation between the variables.

The need to keep these smaller oscillations under control limits the stepsize that can be used. As can be seen in the rightmost plot in Figure 5.3, there are also oscillations in the value of the Hamiltonian (which would be constant if the trajectory were simulated exactly). If a larger stepsize were used, these oscillations would be larger. At a critical stepsize ($\varepsilon = 0.45$ in this example), the trajectory becomes unstable, and the value of the Hamiltonian grows without bound. As long as the stepsize is less than this, however, the error in the Hamiltonian stays bounded regardless of the number of leapfrog steps done. This lack of growth in the error is not guaranteed for all Hamiltonians, but it does hold for many distributions more complex than Gaussians. As can be seen, however, the error in the Hamiltonian along the trajectory does tend to be positive more often than negative. In this example, the error is $+0.41$ at the end of the trajectory, so if this trajectory were used for an HMC proposal, the probability of accepting the endpoint as the next state would be $\exp(-0.41) = 0.66$.

5.3.3.2 Sampling from a Two-Dimensional Distribution

Figures 5.4 and 5.5 show the results of using HMC and a simple random-walk Metropolis method to sample from a bivariate Gaussian similar to the one just discussed, but with stronger correlation of 0.98.

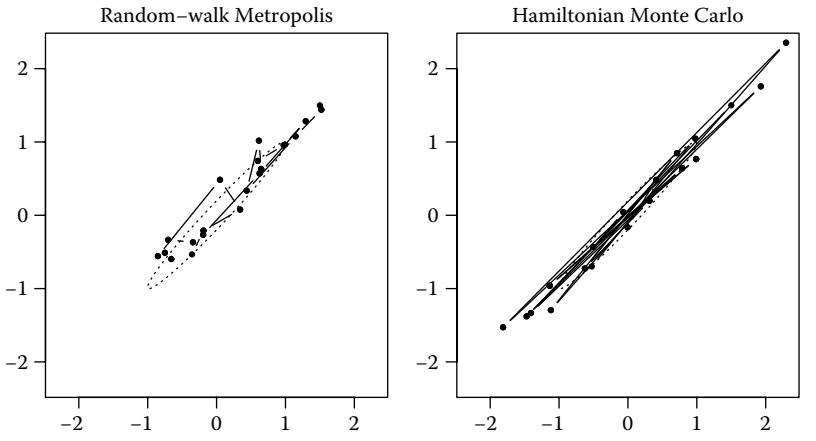

FIGURE 5.4
Twenty iterations of the random-walk Metropolis method (with 20 updates per iteration) and of the Hamiltonian Monte Carlo method (with 20 leapfrog steps per trajectory) for a two-dimensional Gaussian distribution with marginal standard deviations of one and correlation 0.98. Only the two position coordinates are plotted, with ellipses drawn one standard deviation away from the mean.

In this example, as in the previous one, HMC used a kinetic energy (defining the momentum distribution) of $K(p) = p^T p/2$. The results of 20 HMC iterations, using trajectories of $L = 20$ leapfrog steps with stepsize $\varepsilon = 0.18$, are shown in the right plot of Figure 5.4. These values were chosen so that the trajectory length, εL, is sufficient to move to a distant point in the distribution, without being so large that the trajectory will often waste computation time by doubling back on itself. The rejection rate for these trajectories was 0.09.

Figure 5.4 also shows every 20th state from 400 iterations of random-walk Metropolis, with a bivariate Gaussian proposal distribution with the current state as mean, zero correlation, and the same standard deviation for the two coordinates. The standard deviation of the proposals for this example was 0.18, which is the same as the stepsize used for HMC proposals, so that the change in state in these random-walk proposals was comparable to that for a single leapfrog step for HMC. The rejection rate for these random-walk proposals was 0.37.

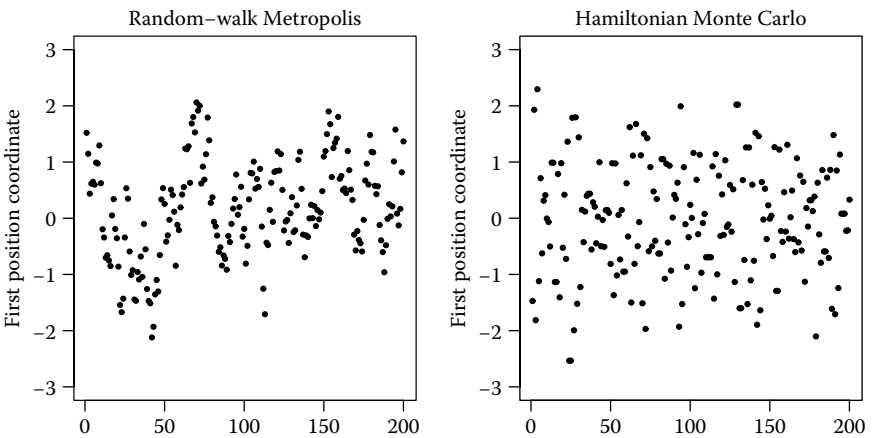

FIGURE 5.5
Two hundred iterations, starting with the 20 iterations shown above, with only the first position coordinate plotted.

One can see in Figure 5.4 how the systematic motion during an HMC trajectory (illustrated in Figure 5.3) produces larger changes in state than a corresponding number of random-walk Metropolis iterations. Figure 5.5 illustrates this difference for longer runs of 20×200 random-walk Metropolis iterations and of 200 HMC iterations.

5.3.3.3 The Benefit of Avoiding Random Walks

Avoidance of random-walk behavior, as illustrated above, is one major benefit of HMC. In this example, because of the high correlation between the two position variables, keeping the acceptance probability for random-walk Metropolis reasonably high requires that the changes proposed have a magnitude comparable to the standard deviation in the most constrained direction (0.14 in this example, the square root of the smallest eigenvalue of the covariance matrix). The changes produced using one Gibbs sampling scan would be of similar magnitude. The number of iterations needed to reach a state almost independent of the current state is mostly determined by how long it takes to explore the less constrained direction, which for this example has standard deviation 1.41—about ten times greater than the standard deviation in the most constrained direction. We might therefore expect that we would need around 10 iterations of random-walk Metropolis in which the proposal was accepted to move to a nearly independent state. But the number needed is actually roughly the square of this—around 100 iterations with accepted proposals—because the random-walk Metropolis proposals have no tendency to move consistently in the same direction.

To see this, note that the variance of the position after n iterations of random-walk Metropolis from some start state will grow in proportion to n (until this variance becomes comparable to the overall variance of the state), since the position is the sum of mostly independent movements for each iteration. The *standard deviation* of the amount moved (which gives the typical amount of movement) is therefore proportional to $\sqrt{n}$.

The stepsize used for the leapfrog steps is similarly limited by the most constrained direction, but the movement will be in the same direction for many steps. The distance moved after n steps will therefore tend to be proportional to n, until the distance moved becomes comparable to the overall width of the distribution. The advantage compared to movement by a random walk will be a factor roughly equal to the ratio of the standard deviations in the least confined direction and most confined direction—about 10 here.

Because avoiding a random walk is so beneficial, the optimal standard deviation for random-walk Metropolis proposals in this example is actually much larger than the value of 0.18 used here. A proposal standard deviation of 2.0 gives a very low acceptance rate (0.06), but this is more than compensated for by the large movement (to a nearly independent point) on the rare occasions when a proposal is accepted, producing a method that is about as efficient as HMC. However, this strategy of making large changes with a small acceptance rate works only when, as here, the distribution is tightly constrained in only one direction.

5.3.3.4 Sampling from a 100-Dimensional Distribution

More typical behavior of HMC and random-walk Metropolis is illustrated by a 100-dimensional multivariate Gaussian distribution in which the variables are independent, with means of zero, and standard deviations of 0.01, 0.02, . . . , 0.99, 1.00. Suppose that we have no knowledge of the details of this distribution, so we will use HMC with the same simple, rotationally symmetric kinetic energy function as above, $K(p) = p^T p/2$, and use random-walk Metropolis proposals in which changes to each variable are independent, all

with the same standard deviation. As discussed below in Section 5.4.1, the performance of both these sampling methods is invariant to rotation, so this example is illustrative of how they perform on any multivariate Gaussian distribution in which the square roots of the eigenvalues of the covariance matrix are $0.01, 0.02, \ldots, 0.99, 1.00$.

For this problem, the position coordinates, q_i, and corresponding momentum coordinates, p_i, are all independent, so the leapfrog steps used to simulate a trajectory operate independently for each (q_i, p_i) pair. However, whether the trajectory is accepted depends on the total error in the Hamiltonian due to the leapfrog discretization, which is a sum of the errors due to each (q_i, p_i) pair (for the terms in the Hamiltonian involving this pair). Keeping this error small requires limiting the leapfrog stepsize to a value roughly equal to the smallest of the standard deviations (0.01), which implies that many leapfrog steps will be needed to move a distance comparable to the largest of the standard deviations (1.00).

Consistent with this, I applied HMC to this distribution using trajectories with $L = 150$ and with ε randomly selected for each iteration, uniformly from $(0.0104, 0.0156)$, which is $0.013 \pm 20\%$. I used random-walk Metropolis with proposal standard deviation drawn uniformly from $(0.0176, 0.0264)$, which is $0.022 \pm 20\%$. These are close to optimal settings for both methods. The rejection rate was 0.13 for HMC and 0.75 for random-walk Metropolis.

Figure 5.6 shows results from runs of 1000 iterations of HMC (right) and of random-walk Metropolis (left), counting 150 random-walk Metropolis updates as one iteration, so that the computation time per iteration is comparable to that for HMC. The plot shows the last variable, with the largest standard deviation. The autocorrelation of these values is clearly much higher for random-walk Metropolis than for HMC. Figure 5.7 shows the estimates for the mean and standard deviation of each of the 100 variables obtained using the HMC and random-walk Metropolis runs (estimates were just the sample means and sample standard deviations of the values from the 1000 iterations). Except for the first few

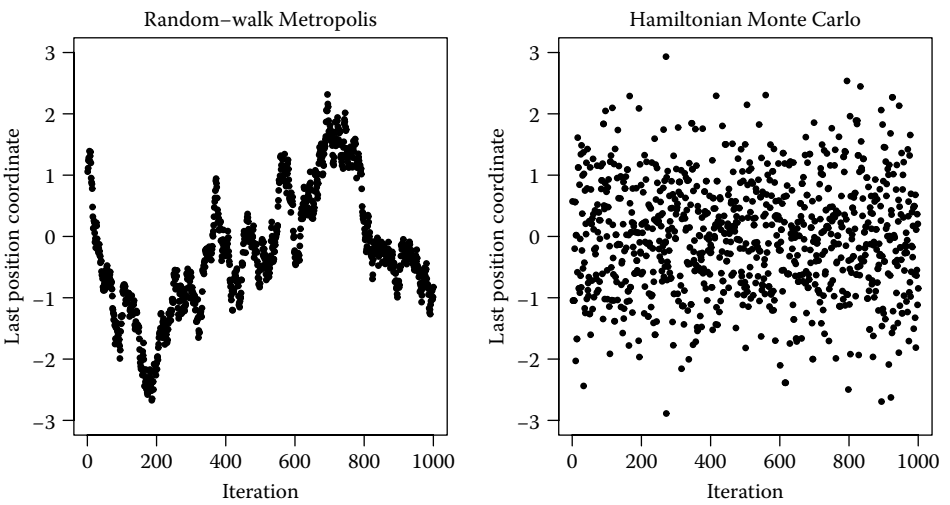

FIGURE 5.6
Values for the variable with largest standard deviation for the 100-dimensional example, from a random-walk Metropolis run and an HMC run with $L = 150$. To match computation time, 150 updates were counted as one iteration for random-walk Metropolis.

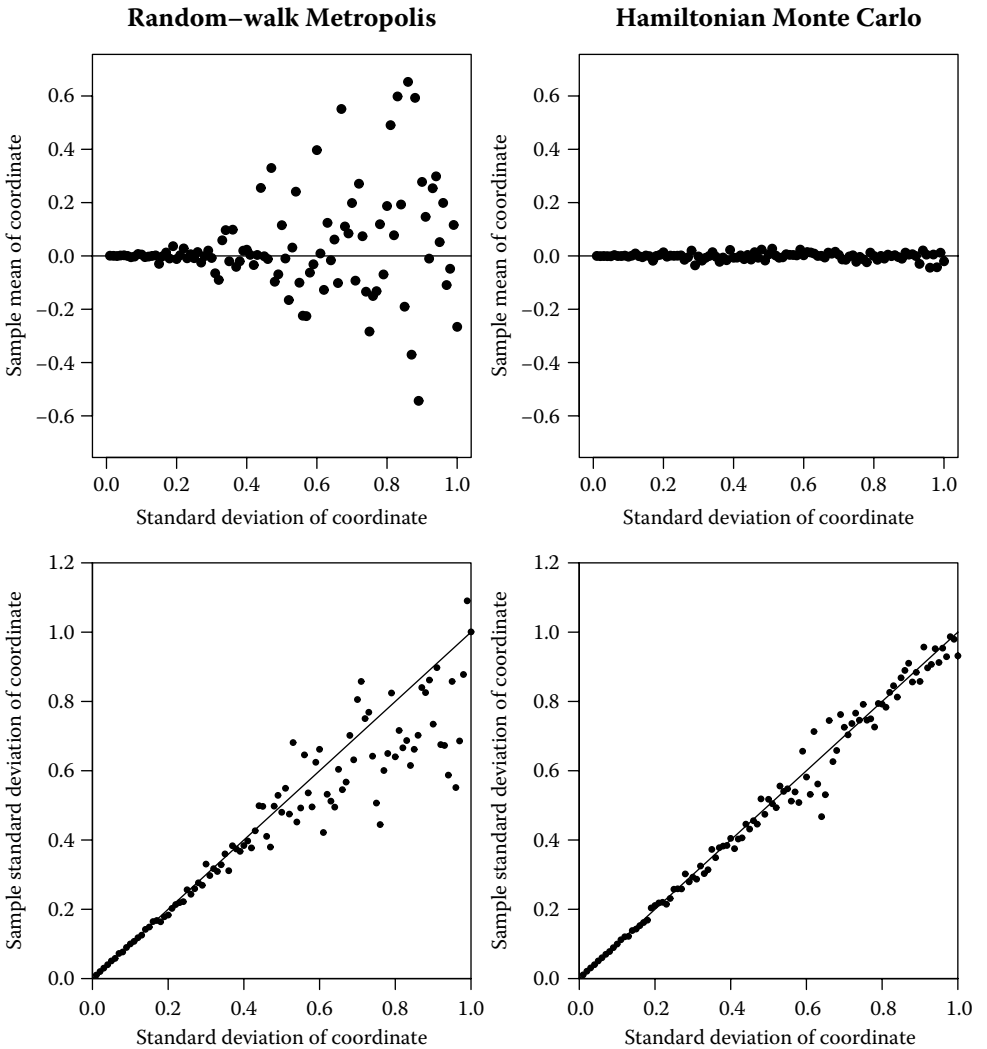

FIGURE 5.7
Estimates of means (top) and standard deviations (bottom) for the 100-dimensional example, using random-walk Metropolis (left) and HMC (right). The 100 variables are labeled on the horizontal axes by the true standard deviaton of that variable. Estimates are on the vertical axes.

variables (with smallest standard deviations), the error in the mean estimates from HMC is roughly 10 times less than the error in the mean estimates from random-walk Metropolis. The standard deviation estimates from HMC are also better.

The randomization of the leapfrog stepsize done in this example follows the advice discussed at the end of Section 5.3.2. In this example, not randomizing the stepsize (fixing $\varepsilon = 0.013$) does in fact cause problems—the variables with standard deviations near 0.31 or 0.62 change only slowly, since 150 leapfrog steps with $\epsilon = 0.013$ produces nearly a full or half cycle for these variables, so an accepted trajectory does not make much of a change in the absolute value of the variable.

5.4 HMC in Practice and Theory

Obtaining the benefits from HMC illustrated in the previous section, including random-walk avoidance, requires proper tuning of L and ε. I discuss tuning of HMC below, and also show how performance can be improved by using whatever knowledge is available regarding the scales of variables and their correlations. After briefly discussing what to do when HMC alone is not enough, I discuss an additional benefit of HMC—its better scaling with dimensionality than simple Metropolis methods.

5.4.1 Effect of Linear Transformations

Like all MCMC methods I am aware of, the performance of HMC may change if the variables being sampled are transformed by multiplication by some nonsingular matrix, A. However, performance stays the same (except perhaps in terms of computation time per iteration) if at the same time the corresponding momentum variables are multiplied by $(A^T)^{-1}$. These facts provide insight into the operation of HMC, and can help us improve performance when we have some knowledge of the scales and correlations of the variables.

Let the new variables be $q' = Aq$. The probability density for q' will be given by $P'(q') = P(A^{-1}q')/|\det(A)|$, where $P(q)$ is the density for q. If the distribution for q is the canonical distribution for a potential energy function $U(q)$ (see Section 5.3.1), we can obtain the distribution for q' as the canonical distribution for $U'(q') = U(A^{-1}q')$. (Since $|\det(A)|$ is a constant, we need not include a $\log|\det(A)|$ term in the potential energy.)

We can choose whatever distribution we wish for the corresponding momentum variables, so we could decide to use the same kinetic energy as before. Alternatively, we can choose to transform the momentum variables by $p' = (A^T)^{-1}p$, and use a new kinetic energy of $K'(p') = K(A^T p')$. If we were using a quadratic kinetic energy, $K(p) = p^T M^{-1} p/2$ (see Equation 5.5), the new kinetic energy will be

$$K'(p') = (A^T p')^T M^{-1}(A^T p')/2 = (p')^T (A M^{-1} A^T)\, p'/2 = (p')^T (M')^{-1} p'/2, \tag{5.25}$$

where $M' = (A M^{-1} A^T)^{-1} = (A^{-1})^T M A^{-1}$.

If we use momentum variables transformed in this way, the dynamics for the new variables, (q', p'), essentially replicates the original dynamics for (q, p), so the performance of HMC will be the same. To see this, note that if we follow Hamiltonian dynamics for (q', p'), the result in terms of the original variables will be as follows (see Equations 5.6 and 5.7):

$$\frac{dq}{dt} = A^{-1}\frac{dq'}{dt} = A^{-1}(M')^{-1}p' = A^{-1}(A M^{-1} A^T)(A^T)^{-1}\, p = M^{-1}\, p,$$

$$\frac{dp}{dt} = A^T\frac{dp'}{dt} = -A^T\,\nabla U'(q') = -A^T\,(A^{-1})^T\,\nabla U(A^{-1}q') = -\nabla U(q),$$

which matches what would happen following Hamiltonian dynamics for (q, p).

If A is an orthogonal matrix (such as a rotation matrix), for which $A^{-1} = A^T$, the performance of HMC is unchanged if we transform both q and p by multiplying by A (since $(A^T)^{-1} = A$). If we chose a rotationally symmetric distribution for the momentum, with $M = mI$ (i.e. the momentum variables are independent, each having variance m), such an orthogonal transformation will not change the kinetic energy function (and hence not change the distribution of the momentum variables), since we will have $M' = (A\,(mI)^{-1}A^T)^{-1} = mI$.

Such an invariance to rotation holds also for a random-walk Metropolis method in which the proposal distribution is rotationally symmetric (e.g. Gaussian with covariance matrix mI). In contrast, Gibbs sampling is not rotationally invariant, nor is a scheme in which the Metropolis algorithm is used to update each variable in turn (with a proposal that changes only that variable). However, Gibbs sampling is invariant to rescaling of the variables (transformation by a diagonal matrix), which is not true for HMC or random-walk Metropolis, unless the kinetic energy or proposal distribution is transformed in a corresponding way.

Suppose that we have an estimate, Σ, of the covariance matrix for q, and suppose also that q has at least a roughly Gaussian distribution. How can we use this information to improve the performance of HMC? One way is to transform the variables so that their covariance matrix is close to the identity, by finding the Cholesky decomposition, $\Sigma = LL^T$, with L being lower-triangular, and letting $q' = L^{-1}q$. We then let our kinetic energy function be $K(p) = p^T p/2$. Since the momentum variables are independent, and the position variables are close to independent with variances close to one (if our estimate Σ and our assumption that q is close to Gaussian are good), HMC should perform well using trajectories with a small number of leapfrog steps, which will move all variables to a nearly independent point. More realistically, the estimate Σ may not be very good, but this transformation could still improve performance compared to using the same kinetic energy with the original q variables.

An equivalent way to make use of the estimated covariance Σ is to keep the original q variables, but use the kinetic energy function $K(p) = p^T \Sigma p/2$—that is, we let the momentum variables have covariance Σ^{-1}. The equivalence can be seen by transforming this kinetic energy to correspond to a transformation to $q' = L^{-1}q$ (see Equation 5.25), which gives $K(p') = (p')^T M'^{-1} p'$ with $M' = (L^{-1}(LL^T)(L^{-1})^T)^{-1} = I$.

Using such a kinetic energy function to compensate for correlations between position variables has a long history in molecular dynamics (Bennett, 1975). The usefulness of this technique is limited by the computational cost of matrix operations when the dimensionality is high.

Using a diagonal Σ can be feasible even in high-dimensional problems. Of course, this provides information only about the different scales of the variables, not their correlation. Moreover, when the actual correlations are nonzero, it is not clear what scales to use. Making an optimal choice is probably infeasible. Some approximation to the conditional standard deviation of each variable given all the others may be possible—as I have done for Bayesian neural network models (Neal, 1996a). If this also is not feasible, using approximations to the marginal standard deviations of the variables may be better than using the same scale for them all.

5.4.2 Tuning HMC

One practical impediment to the use of Hamiltonian Monte Carlo is the need to select suitable values for the leapfrog stepsize, ε, and the number of leapfrog steps, L, which together determine the length of the trajectory in fictitious time, εL. Most MCMC methods have parameters that need to be tuned, with the notable exception of Gibbs sampling when the conditional distributions are amenable to direct sampling. However, tuning HMC is more difficult in some respects than tuning a simple Metropolis method.

5.4.2.1 Preliminary Runs and Trace Plots

Tuning HMC will usually require preliminary runs with trial values for ε and L. In judging how well these runs work, trace plots of quantities that are thought to be indicative

of overall convergence should be examined. For Bayesian inference problems, high-level hyperparameters are often among the slowest-moving quantities. The value of the potential energy function, $U(q)$, is also usually of central significance. The autocorrelation for such quantities indicates how well the Markov chain is exploring the state space. Ideally, we would like the state after one HMC iteration to be nearly independent of the previous state.

Unfortunately, preliminary runs can be misleading, if they are not long enough to have reached equilibrium. It is possible that the best choices of ε and L for reaching equilibrium are different from the best choices once equilibrium is reached, and even at equilibrium, it is possible that the best choices vary from one place to another. If necessary, at each iteration of HMC, ε and L can be chosen randomly from a selection of values that are appropriate for different parts of the state space (or these selections and can be used sequentially).

Doing several runs with different random starting states is advisable (for both preliminary and final runs), so that problems with isolated modes can be detected. Note that HMC is no less (or more) vulnerable to problems with isolated modes than other MCMC methods that make local changes to the state. If isolated modes are found to exist, something needs to be done to solve this problem—just combining runs that are each confined to a single mode is not valid. A modification of HMC with "tempering" along a trajectory (Section 5.5.7) can sometimes help with multiple modes.

5.4.2.2 What Stepsize?

Selecting a suitable leapfrog stepsize, ε, is crucial. Too large a stepsize will result in a very low acceptance rate for states proposed by simulating trajectories. Too small a stepsize will either waste computation time, by the same factor as the stepsize is too small, or (worse) will lead to slow exploration by a random walk, if the trajectory length, εL, is then too short (i.e. L is not large enough; see below).

Fortunately, as illustrated in Figure 5.3, the choice of stepsize is almost independent of how many leapfrog steps are done. The error in the value of the Hamiltonian (which will determine the rejection rate) usually does not increase with the number of leapfrog steps, *provided* that the stepsize is small enough that the dynamics is stable.

The issue of stability can be seen in a simple one-dimensional problem in which the following Hamiltonian is used:

$$H(q,p) = \frac{q^2}{2\sigma^2} + \frac{p^2}{2}.$$

The distribution for q that this defines is Gaussian with standard deviation σ. A leapfrog step for this system (as for any quadratic Hamiltonian) will be a linear mapping from $(q(t), p(t))$ to $(q(t+\varepsilon), p(t+\varepsilon))$. Referring to Equations 5.18 through 5.20, we see that this mapping can be represented by a matrix multiplication as follows:

$$\begin{bmatrix} q(t+\varepsilon) \\ p(t+\varepsilon) \end{bmatrix} = \begin{bmatrix} 1 - \varepsilon^2/2\sigma^2 & \varepsilon \\ -\varepsilon/\sigma^2 + \varepsilon^3/4\sigma^4 & 1 - \varepsilon^2/2\sigma^2 \end{bmatrix} \begin{bmatrix} q(t) \\ p(t) \end{bmatrix}.$$

Whether iterating this mapping leads to a stable trajectory, or one that diverges to infinity, depends on the magnitudes of the eigenvalues of the above matrix, which are

$$\left(1 - \frac{\varepsilon^2}{2\sigma^2}\right) \pm \left(\frac{\varepsilon}{\sigma}\right)\sqrt{\varepsilon^2/4\sigma^2 - 1}.$$

When $\varepsilon/\sigma > 2$, these eigenvalues are real, and at least one will have absolute value greater than one. Trajectories computed using the leapfrog method with this ε will therefore be unstable. When $\varepsilon/\sigma < 2$, the eigenvalues are complex, and both have squared magnitude of

$$\left(1 - \frac{\varepsilon^2}{2\sigma^2}\right)^2 + \left(\frac{\varepsilon^2}{\sigma^2}\right)\left(1 - \frac{\varepsilon^2}{4\sigma^2}\right) = 1.$$

Trajectories computed with $\varepsilon < 2\sigma$ are therefore stable.

For multidimensional problems in which the kinetic energy used is $K(p) = p^T p/2$ (as in the example above), the stability limit for ε will be determined (roughly) by the width of the distribution in the most constrained direction—for a Gaussian distribution, this would the square root of the smallest eigenvalue of the covariance matrix for q. Stability for more general quadratic Hamiltonians with $K(p) = p^T M^{-1} p/2$ can be determined by applying a linear transformation that makes $K(p') = (p')^T p'/2$, as discussed above in Section 5.4.1.

When a stepsize, ε, that produces unstable trajectories is used, the value of H grows exponentially with L, and consequently the acceptance probability will be extremely small. For low-dimensional problems, using a value for ε that is just a little below the stability limit is sufficient to produce a good acceptance rate. For high-dimensional problems, however, the stepsize may need to be reduced further than this to keep the error in H to a level that produces a good acceptance probability. This is discussed further in Section 5.4.4.

Choosing too large a value of ε can have very bad effects on the performance of HMC. In this respect, HMC is more sensitive to tuning than random-walk Metropolis. A standard deviation for proposals needs to be chosen for random-walk Metropolis, but performance degrades smoothly as this choice is made too large, without the sharp degradation seen with HMC when ε exceeds the stability limit. (However, in high-dimensional problems, the degradation in random-walk Metropolis with too large a proposal standard deviation can also be quite sharp, so this distinction becomes less clear.)

This sharp degradation in performance of HMC when the stepsize is too big would not be a serious issue if the stability limit were constant—the problem would be obvious from preliminary runs, and so could be fixed. The real danger is that the stability limit may differ for several regions of the state space that all have substantial probability. If the preliminary runs are started in a region where the stability limit is large, a choice of ε a little less than this limit might appear to be appropriate. However, if this ε is above the stability limit for some other region, the runs may never visit this region, even though it has substantial probability, producing a drastically wrong result. To see why this could happen, note that if the run ever does visit the region where the chosen ε would produce instability, it will stay there for a very long time, since the acceptance probability with that ε will be very small. Since the method nevertheless leaves the correct distribution invariant, it follows that the run only rarely moves to this region from a region where the chosen ε leads to stable trajectories. One simple context where this problem can arise is when sampling from a distribution with very light tails (lighter than a Gaussian distribution), for which the log of the density will fall faster than quadratically. In the tails, the gradient of the log density will be large, and a small stepsize will be needed for stability. See Roberts and Tweedie (1996) for a discussion of this in the context of the Langevin method (see Section 5.5.2).

This problem can be alleviated by choosing ε randomly from some distribution. Even if the mean of this distribution is too large, suitably small values for ε may be chosen occasionally. (See Section 5.3.2 for another reason to randomly vary the stepsize.) The random choice of ε should be done once at the start of a trajectory, not for every leapfrog step, since even if

all the choices are below the stability limit, random changes at each step lead to a random walk in the error for H, rather than the bounded error that is illustrated in Figure 5.3.

The "short-cut" procedures described in Section 5.5.6 can be seen as ways of saving computation time when a randomly chosen stepsize is inappropriate.

5.4.2.3 What Trajectory Length?

Choosing a suitable trajectory length is crucial if HMC is to explore the state space systematically, rather than by a random walk. Many distributions are difficult to sample from because they are tightly constrained in some directions, but much less constrained in other directions. Exploring the less constrained directions is best done using trajectories that are long enough to reach a point that is far from the current point in that direction. Trajectories can be too long, however, as is illustrated in Figure 5.3. The trajectory shown on the left of that figure is a bit too long, since it reverses direction and then ends at a point that might have been reached with a trajectory about half its length. If the trajectory were a little longer, the result could be even worse, since the trajectory would not only take longer to compute, but might also end near its starting point.

For more complex problems, one cannot expect to select a suitable trajectory length by looking at plots like Figure 5.3. Finding the linear combination of variables that is least confined will be difficult, and will be impossible when, as is typical, the least confined "direction" is actually a nonlinear curve or surface.

Setting the trajectory length by trial and error therefore seems necessary. For a problem thought to be fairly difficult, a trajectory with $L = 100$ might be a suitable starting point. If preliminary runs (with a suitable ε; see above) show that HMC reaches a nearly independent point after only one iteration, a smaller value of L might be tried next. (Unless these "preliminary" runs are actually sufficient, in which case there is of course no need to do more runs.) If instead there is high autocorrelation in the run with $L = 100$, runs with $L = 1000$ might be tried next.

As discussed at the end of Sections 5.3.2 and 5.3.3, randomly varying the length of the trajectory (over a fairly small interval) may be desirable, to avoid choosing a trajectory length that happens to produce a near-periodicity for some variable or combination of variables.

5.4.2.4 Using Multiple Stepsizes

Using the results in Section 5.4.1, we can exploit information about the relative scales of variables to improve the performance of HMC. This can be done in two equivalent ways. If s_i is a suitable scale for q_i, we could transform q, by setting $q_i' = q_i/s_i$, or we could instead use a kinetic energy function of $K(p) = p^T M^{-1} p$, with M being a diagonal matrix with diagonal elements $m_i = 1/s_i^2$.

A third equivalent way to exploit this information, which is often the most convenient, is to use different stepsizes for different pairs of position and momentum variables. To see how this works, consider a leapfrog update (following Equations 5.18 through 5.20) with $m_i = 1/s_i^2$:

$$p_i\,(t + \varepsilon/2) = p_i(t) - (\varepsilon/2)\ \frac{\partial U}{\partial q_i}(q(t)),$$

$$q_i(t + \varepsilon) = q_i(t) + \varepsilon\, s_i^2\, p_i\,(t + \varepsilon/2)\,,$$

$$p_i(t + \varepsilon) = p_i\,(t + \varepsilon/2) - (\varepsilon/2)\ \frac{\partial U}{\partial q_i}(q(t + \varepsilon)).$$

Define $(q^{(0)}, p^{(0)})$ to be the state at the beginning of the leapfrog step (i.e. $(q(t), p(t))$), define $(q^{(1)}, p^{(1)})$ to be the final state (i.e. $(q(t+\varepsilon), p(t+\varepsilon))$), and define $p^{(1/2)}$ to be half-way momentum (i.e. $p(t+\varepsilon/2)$). We can now rewrite the leapfrog step above as

$$\begin{aligned} p_i^{(1/2)} &= p_i^{(0)} - (\varepsilon/2)\,\frac{\partial U}{\partial q_i}(q^{(0)}), \\ q_i^{(1)} &= q_i^{(0)} + \varepsilon\, s_i^2\, p_i^{(1/2)}, \\ p_i^{(1)} &= p_i^{(1/2)} - (\varepsilon/2)\,\frac{\partial U}{\partial q_i}(q^{(1)}). \end{aligned}$$

If we now define rescaled momentum variables, $\tilde{p}_i = s_i p_i$, and stepsizes $\varepsilon_i = s_i \varepsilon$, we can write the leapfrog update as

$$\begin{aligned} \tilde{p}_i^{(1/2)} &= \tilde{p}_i^{(0)} - (\varepsilon_i/2)\,\frac{\partial U}{\partial q_i}(q^{(0)}), \\ q_i^{(1)} &= q_i^{(0)} + \varepsilon_i\, \tilde{p}_i^{(1/2)}, \\ \tilde{p}_i^{(1)} &= \tilde{p}_i^{(1/2)} - (\varepsilon_i/2)\,\frac{\partial U}{\partial q_i}(q^{(1)}). \end{aligned}$$

This is just like a leapfrog update with all $m_i = 1$, but with different stepsizes for different (q_i, p_i) pairs. Of course, the successive values for $(q, \tilde{p})$ can no longer be interpreted as following Hamiltonian dynamics at consistent time points, but that is of no consequence for the use of these trajectories in HMC. Note that when we sample for the momentum before each trajectory, each $\tilde{p}_i$ is drawn independently from a Gaussian distribution with mean zero and variance one, regardless of the value of s_i.

This multiple stepsize approach is often more convenient, especially when the estimated scales, s_i, are not fixed, as discussed in Section 5.4.5, and the momentum is only partially refreshed (Section 5.5.3).

5.4.3 Combining HMC with Other MCMC Updates

For some problems, MCMC using HMC alone will be impossible or undesirable. Two situations where non-HMC updates will be necessary are when some of the variables are discrete, and when the derivatives of the log probability density with respect to some of the variables are expensive or impossible to compute. HMC can then be feasibly applied only to the other variables. Another example is when special MCMC updates have been devised that may help convergence in ways that HMC does not—for example, by moving between otherwise isolated modes—but which are not a complete replacement for HMC. As discussed in Section 5.4.5 below, Bayesian hierarchical models may also be best handled with a combination of HMC and other methods such as Gibbs sampling.

In such circumstances, one or more HMC updates for all or a subset of the variables can be alternated with one or more other updates that leave the desired joint distribution of all variables invariant. The HMC updates can be viewed as either leaving this same joint distribution invariant, or as leaving invariant the conditional distribution of the variables that HMC changes, given the current values of the variables that are fixed during the HMC update. These are equivalent views, since the joint density can be factored as this conditional density times the marginal density of the variables that are fixed, which is just a constant

from the point of view of a single HMC update, and hence can be left out of the potential energy function.

When both HMC and other updates are used, it may be best to use shorter trajectories for HMC than would be used if only HMC were being done. This allows the other updates to be done more often, which presumably helps sampling. Finding the optimal tradeoff is likely to be difficult, however. A variation on HMC that reduces the need for such a tradeoff is described below in Section 5.5.3.

5.4.4 Scaling with Dimensionality

In Section 5.3.3, one of the main benefits of HMC was illustrated—its ability to avoid the inefficient exploration of the state space via a random walk. This benefit is present (to at least some degree) for most practical problems. For problems in which the dimensionality is moderate to high, another benefit of HMC over simple random-walk Metropolis methods is a slower increase in the computation time needed (for a given level of accuracy) as the dimensionality increases. (Note that here I will consider only sampling performance after equilibrium is reached, not the time needed to approach equilibrium from some initial state not typical of the distribution, which is harder to analyze.)

5.4.4.1 Creating Distributions of Increasing Dimensionality by Replication

To talk about how performance scales with dimensionality we need to assume something about how the distribution changes with dimensionality, d.

I will assume that dimensionality increases by adding independent replicas of variables—that is, the potential energy function for $q = (q_1, \ldots, q_d)$ has the form $U(q) = \Sigma\, u_i(q_i)$, for functions u_i drawn independently from some distribution. Of course, this is not what any real practical problem is like, but it may be a reasonable model of the effect of increasing dimensionality for some problems—for instance, in statistical physics, distant regions of large systems are often nearly independent. Note that the independence assumption itself is not crucial since, as discussed in Section 5.4.1, the performance of HMC (and of simple random-walk Metropolis) does not change if independence is removed by rotating the coordinate system, provided the kinetic energy function (or random-walk proposal distribution) is rotationally symmetric.

For distributions of this form, in which the variables are independent, Gibbs sampling will perform very well (assuming it is feasible), producing an independent point after each scan of all variables. Applying Metropolis updates to each variable separately will also work well, provided the time for a single-variable update does not grow with d. However, these methods are not invariant to rotation, so this good performance may not generalize to the more interesting distributions for which we hope to obtain insight with the analysis below.

5.4.4.2 Scaling of HMC and Random-Walk Metropolis

Here, I discuss informally how well HMC and random-walk Metropolis scale with dimension, loosely following Creutz (1988, Section III).

To begin, Cruetz notes that the following relationship holds when any Metropolis-style algorithm is used to sample a density $P(x) = (1/Z)\exp(-E(x))$:

$$1 = \mathrm{E}\,[P(x^*)/P(x)] = \mathrm{E}\,[\exp(-(E(x^*) - E(x)))] = \mathrm{E}\,[\exp(-\Delta)], \tag{5.26}$$

where x is the current state, assumed to be distributed according to $P(x)$, x^* is the proposed state, and $\Delta = E(x^*) - E(x)$. Jensen's inequality then implies that the expectation of the energy difference is nonnegative:

$$\mathrm{E}[\Delta] \geq 0.$$

The inequality will usually be strict.

When $U(q) = \Sigma\, u_i(q_i)$, and proposals are produced independently for each i, we can apply these relationships either to a single variable (or pair of variables) or to the entire state. For a single variable (or pair), I will write Δ_1 for $E(x^*) - E(x)$, with $x = q_i$ and $E(x) = u_i(q_i)$, or $x = (q_i, p_i)$ and $E(x) = u_i(q_i) + p_i^2/2$. For the entire state, I will write Δ_d for $E(x^*) - E(x)$, with $x = q$ and $E(x) = U(q)$, or $x = (q, p)$ and $E(x) = U(q) + K(p)$. For both random-walk Metropolis and HMC, increasing dimension by replicating variables will lead to increasing energy differences, since Δ_d is the sum of Δ_1 for each variable, each of which has positive mean. This will lead to a decrease in the acceptance probability—equal to $\min(1, \exp(-\Delta_d))$—unless the width of the proposal distribution or the leapfrog stepsize is decreased to compensate.

More specifically, for random-walk Metropolis with proposals that change each variable independently, the difference in potential energy between a proposed state and the current state will be the sum of independent differences for each variable. If we fix the standard deviation, ς, for each proposed change, the mean and the variance of this potential energy difference will both grow linearly with d, which will lead to a progressively lower acceptance rate. To maintain reasonable performance, ς will have to decrease as d increases. Furthermore, the number of iterations needed to reach a nearly independent point will be proportional to ς^{-2}, since exploration is via a random walk.

Similarly, when HMC is used to sample from a distribution in which the components of q are independent, using the kinetic energy $K(p) = \Sigma\, p_i^2/2$, the different (q_i, p_i) pairs do not interact during the simulation of a trajectory—each (q_i, p_i) pair follows Hamiltonian dynamics according to just the one term in the potential energy involving q_i and the one term in the kinetic energy involving p_i. There is therefore no need for the length in fictitious time of a trajectory to increase with dimensionality. However, acceptance of the endpoint of the trajectory is based on the error in H due to the leapfrog approximation, which is the sum of the errors pertaining to each (q_i, p_i) pair. For a fixed stepsize, ε, and fixed trajectory length, εL, both the mean and the variance of the error in H grow linearly with d. This will lead to a progressively lower acceptance rate as dimensionality increases, if it is not counteracted by a decrease in ε. The number of leapfrog steps needed to reach an independent point will be proportional to ε^{-1}.

To see which method scales better, we need to determine how rapidly we must reduce ς and ε as d increases, in order to maintain a reasonable acceptance rate. As d increases and ς or ε goes to zero, Δ_1 will go to zero as well. Using a second-order approximation of $\exp(-\Delta_1)$ as $1 - \Delta_1 + \Delta_1^2/2$, together with Equation 5.26, we find that

$$\mathrm{E}[\Delta_1] \approx \frac{\mathrm{E}[\Delta_1^2]}{2}. \tag{5.27}$$

It follows from this that the variance of Δ_1 is twice the mean of Δ_1 (when Δ_1 is small), which implies that the variance of Δ_d is twice the mean of Δ_d (even when Δ_d is not small). To achieve a good acceptance rate, we must therefore keep the mean of Δ_d near one, since a large mean will not be saved by a similarly large standard deviation (which would produce fairly frequent acceptances as Δ_d occasionally takes on a negative value).

For random-walk Metropolis with a symmetric proposal distribution, we can see how ς needs to scale by directly averaging Δ_1 for a proposal and its inverse. Let the proposal for

one variable be $x^* = x + c$, and suppose that $c = a$ and $c = -a$ are equally likely. Approximating $U(x^*)$ to second order as $U(x) + cU'(x) + c^2U''(x)/2$, we find that the average of $\Delta_1 = U(x^*) - U(x)$ over $c = a$ and $c = -a$ is $a^2U''(x)$. Averaging this over the distribution of a, with standard deviation ς, and over the distribution of x, we see that $\mathrm{E}[\Delta_1]$ is proportional to ς^2. It follows that $\mathrm{E}[\Delta_d]$ is proportional to $d\varsigma^2$, so we can maintain a reasonable acceptance rate by letting ς be proportional to $d^{-1/2}$. The number of iterations needed to reach a nearly independent point will be proportional to ς^{-2}, which will be proportional to d. The amount of computation time needed will typically be proportional to d^2.

As discussed at the end of Section 5.2.3, the error in H when using the leapfrog discretization to simulate a trajectory of a fixed length is proportional to ε^2 (for sufficiently small ε). The error in H for a single (q_i, p_i) pair is the same as Δ_1, so we see that Δ_1^2 is proportional to ε^4. Equation 5.27 then implies that $\mathrm{E}[\Delta_1]$ is also proportional to ε^4. The average total error in H for all variables, $\mathrm{E}[\Delta_d]$, will be proportional to $d\varepsilon^4$, and hence we must make ε be proportional to $d^{-1/4}$ to maintain a reasonable acceptance rate. The number of leapfrog updates to reach a nearly independent point will therefore grow as $d^{1/4}$, and the amount of computation time will typically grow as $d^{5/4}$, which is much better than the d^2 growth for random-walk Metropolis.

5.4.4.3 Optimal Acceptance Rates

By extending the analysis above, we can determine what the acceptance rate of proposals is when the optimal choice of ς or ϵ is used. This is helpful when tuning the algorithms—provided, of course, that the distribution sampled is high-dimensional, and has properties that are adequately modeled by a distribution with replicated variables.

To find this acceptance rate, we first note that since Metropolis methods satisfy detailed balance, the probability of an accepted proposal with Δ_d negative must be equal to the probability of an accepted proposal with Δ_d positive. Since all proposals with negative Δ_d are accepted, the acceptance rate is simply twice the probability that a proposal has a negative Δ_d. For large d, the central limit theorem implies that the distribution of Δ_d is Gaussian, since it is a sum of d independent Δ_1 values. (This assumes that the variance of each Δ_1 is finite.) We saw above that the variance of Δ_d is twice its mean, $\mathrm{E}[\Delta_d] = \mu$. The acceptance probability can therefore be written as follows (Gupta et al., 1990), for large d:

$$P(\text{accept}) = 2\,\Phi\left(\frac{(0-\mu)}{\sqrt{2\mu}}\right) = 2\,\Phi\left(-\sqrt{\mu/2}\right) = a(\mu), \tag{5.28}$$

where $\Phi(z)$ is the cumulative distribution function for a Gaussian variable with mean zero and variance one.

For random-walk Metropolis, the cost of obtaining an independent point will be proportional to $1/(a\varsigma^2)$, where a is the acceptance rate. We saw above that $\mu = \mathrm{E}[\Delta_d]$ is proportional to ς^2, so the cost follows the proportionality

$$C_{\mathrm{rw}} \propto \frac{1}{(a(\mu)\mu)}.$$

Numerical calculation shows that this is minimized when $\mu = 2.8$ and $a(\mu) = 0.23$.

For HMC, the cost of obtaining an independent point will be proportional to $1/(a\varepsilon)$, and as we saw above, μ is proportional to ε^4. From this we obtain

$$C_{\text{HMC}} \propto \frac{1}{(a(\mu)\mu^{1/4})}.$$

Numerical calculation shows that the minimum is when $\mu = 0.41$ and $a(\mu) = 0.65$.

The same optimal 23% acceptance rate for random-walk Metropolis was previously obtained using a more formal analysis by Roberts et al. (1997). The optimal 65% acceptance rate for HMC that I derive above is consistent with previous empirical results on distributions following the model here (Neal, 1994, Figure 2), and on real high-dimensional problems (Creutz, 1988, Figures 2 and 3; Sexton and Weingarten, 1992, Table 1). Kennedy and Pendleton (1991) obtained explicit and rigorous results for HMC applied to multivariate Gaussian distributions.

5.4.4.4 Exploring the Distribution of Potential Energy

The better scaling behavior of HMC seen above depends crucially on the resampling of momentum variables. We can see this by considering how well the methods explore the distribution of the potential energy, $U(q) = \Sigma\, u_i(q_i)$. Because $U(q)$ is a sum of d independent terms, its standard deviation will grow in proportion to $d^{1/2}$.

Following Caracciolo et al. (1994), we note that the expected change in potential energy from a single Metropolis update will be no more than order 1—intuitively, large upwards changes are unlikely to be accepted, and since Metropolis updates satisfy detailed balance, large downward changes must also be rare (in equilibrium). Because changes in U will follow a random walk (due again to detailed balance), it will take at least order $(d^{1/2}/1)^2 = d$ Metropolis updates to explore the distribution of U.

In the first step of an HMC iteration, the resampling of momentum variables will typically change the kinetic energy by an amount that is proportional to $d^{1/2}$, since the kinetic energy is also a sum of d independent terms, and hence has standard deviation that grows as $d^{1/2}$ (more precisely, its standard deviation is $d^{1/2}/2^{1/2}$). If the second step of HMC proposes a distant point, this change in kinetic energy (and hence in H) will tend, by the end of the trajectory, to have become equally split between kinetic and potential energy. If the endpoint of this trajectory is accepted, the change in potential energy from a single HMC iteration will be proportional to $d^{1/2}$, comparable to its overall range of variation. So, in contrast to random-walk Metropolis, we may hope that only a few HMC iterations will be sufficient to move to a nearly independent point, even for high-dimensional distributions.

Analyzing how well methods explore the distribution of U can also provide insight into their performance on distributions that are not well modeled by replication of variables, as we will see in the next section.

5.4.5 HMC for Hierarchical Models

Many Bayesian models are defined hierarchically. A large set of low-level parameters have prior distributions that are determined by fewer higher-level "hyperparameters," which in turn may have priors determined by yet-higher-level hyperparameters. For example, in a regression model with many predictor variables, the regression coefficients might be given Gaussian prior distributions, with a mean of zero and a variance that is a hyperparameter. This hyperparameter could be given a broad prior distribution, so that its posterior distribution is determined mostly by the data.

One could apply HMC to these models in an obvious way (after taking the logs of variance hyperparameters, so they will be unconstrained). However, it may be better to apply HMC only to the lower-level parameters, for reasons I will now discuss. (See Section 5.4.3 for general discussion of applying HMC to a subset of variables.)

I will use my work on Bayesian neural network models (Neal, 1996a) as an example. Such models typically have several groups of low-level parameters, each with an associated variance hyperparameter. The posterior distribution of these hyperparameters reflects important aspects of the data, such as which predictor variables are most relevant to the task. The efficiency with which values for these hyperparameters are sampled from the posterior distribution can often determine the overall efficiency of the MCMC method.

I use HMC only for the low-level parameters in Bayesian neural network models, with the hyperparameters being fixed during an HMC update. These HMC updates alternate with Gibbs sampling updates of the hyperparameters, which (in the simpler versions of the models) are independent given the low-level parameters, and have conditional distributions of standard form. By using HMC only for the low-level parameters, the leapfrog stepsizes used can be set using heuristics that are based on the current hyperparameter values. (I use the multiple stepsize approach described at the end of Section 5.4.2, equivalent to using different mass values, m_i, for different parameters.) For example, the size of the network "weights" on connections out of a "hidden unit" determine how sensitive the likelihood function is to changes in weights on connections into the hidden unit; the variance of the weights on these outgoing connections is therefore useful in setting the stepsize for the weights on the incoming connections. If the hyperparameters were changed by the same HMC updates as change the lower-level parameters, using them to set stepsizes would not be valid, since a reversed trajectory would use different stepsizes, and hence not retrace the original trajectory. Without a good way to set stepsizes, HMC for the low-level parameters would likely be much less efficient.

Choo (2000) bypassed this problem by using a modification of HMC in which trajectories are simulated by alternating leapfrog steps that update only the hyperparameters with leapfrog steps that update only the low-level parameters. This procedure maintains both reversibility and volume-preservation (though not necessarily symplecticness), while allowing the stepsizes for the low-level parameters to be set using the current values of the hyperparameters (and vice versa). However, performance did not improve as hoped because of a second issue with hierarchical models.

In these Bayesian neural network models, and many other hierarchical models, the joint distribution of both low-level parameters and hyperparameters is highly skewed, with the probability density varying hugely from one region of high posterior probability to another. Unless the hyperparameters controlling the variances of low-level parameters have very narrow posterior distributions, the joint posterior density for hyperparameters and low-level parameters will vary greatly from when the variance is low to when it is high.

For instance, suppose that in its region of high posterior probability, a variance hyperparameter varies by a factor of 4. If this hyperparameter controls 1000 low-level parameters, their typical prior probability density will vary by a factor of $2^{1000} = 1.07 \times 10^{301}$, corresponding to a potential energy range of $\log(2^{1000}) = 693$, with a standard deviation of $693/12^{1/2} = 200$ (since the variance of a uniform distribution is one twelfth of its range). As discussed at the end of Section 5.4.4, one HMC iteration changes the energy only through the resampling of the momentum variables, which at best leads to a change in potential energy with standard deviation of about $d^{1/2}/2^{3/2}$. For this example, with 1000 low-level

parameters, this is 11.2, so about $(200/11.2)^2 = 319$ HMC iterations will be needed to reach an independent point.

One might obtain similar performance for this example using Gibbs sampling. However, for neural network models, there is no feasible way of using Gibbs sampling for the posterior distribution of the low-level parameters, but HMC can be applied to the conditional distribution of the low-level parameters given the hyperparameters. Gibbs sampling can then be used to update the hyperparameters. As we have seen, performance would not be improved by trying to update the hyperparameters with HMC as well, and updating them by Gibbs sampling is easier.

Choo (2000) tried another approach that could potentially improve on this—reparameterizing low-level parameters θ_i, all with variance $\exp(\kappa)$, by letting $\theta_i = \phi_i \exp(\kappa/2)$, and then sampling for κ and the ϕ_i using HMC. The reparameterization eliminates the extreme variation in probability density that HMC cannot efficiently sample. However, he found that it is difficult to set a suitable stepsize for κ, and that the error in H tended to grow with trajectory length, unlike the typical situation when HMC is used only for the low-level parameters. Use of "tempering" techniques (see Section 5.5.7) is another possibility.

Even though it does not eliminate all difficulties, HMC is very useful for Bayesian neural network models—indeed, without it, they might not be feasible for most applications. Using HMC for at least the low-level parameter can produce similar benefits for other hierarchical models (e.g. Ishwaran, 1999), especially when the posterior correlations of these low-level parameters are high. As in any application of HMC, however, careful tuning of the stepsize and trajectory length is generally necessary.

5.5 Extensions of and Variations on HMC

The basic HMC algorithm (Figure 5.2) can be modified in many ways, either to improve its efficiency, or to make it useful for a wider range of distributions. In this section, I will start by discussing alternatives to the leapfrog discretization of Hamilton's equations, and also show how HMC can handle distributions with constraints on the variables (e.g. variables that must be positive). I will then discuss a special case of HMC—when only one leapfrog step is done—and show how it can be extended to produce an alternative method of avoiding random walks, which may be useful when not all variables are updated by HMC. Most applications of HMC can benefit from using a variant in which "windows" of states are used to increase the acceptance probability. Another widely applicable technique is to use approximations to the Hamiltonian to compute trajectories, while still obtaining correct results by using the exact Hamiltonian when deciding whether to accept the endpoint of the trajectory. Tuning of HMC may be assisted by using a "short-cut" method that avoids computing the whole trajectory when the stepsize is inappropriate. Tempering methods have potential to help with distributions having multiple modes, or which are highly skewed.

There are many other variations that I will not be able to review here, such as the use of a "shadow Hamiltonian" that is exactly conserved by the inexact simulation of the real Hamiltonian (Izagguirre and Hampton, 2004), and the use of symplectic integration methods more sophisticated than the leapfrog method (e.g. Creutz and Gocksch, 1989), including a recent proposal by Girolami et al. (2009) to use a symplectic integrator for a nonseparable

Hamiltonian in which M in the kinetic energy of (Equation 5.5) depends on q, allowing for "adaptation" based on local information.

5.5.1 Discretization by Splitting: Handling Constraints and Other Applications

The leapfrog method is not the only discretization of Hamilton's equations that is reversible and volume-preserving, and hence can be used for HMC. Many "symplectic integration methods" have been devised, mostly for applications other than HMC (e.g. simulating the solar system for millions of years to test its stability). It is possible to devise methods that have a higher order of accuracy than the leapfrog method (see, e.g. McLachlan and Atela, 1992). Using such a method for HMC will produce asymptotically better performance than the leapfrog method, as dimensionality increases. Experience has shown, however, that the leapfrog method is hard to beat in practice.

Nevertheless, it is worth taking a more general look at how Hamiltonian dynamics can be simulated, since this also points to how constraints on the variables can be handled, as well as possible improvements such as exploiting partial analytic solutions.

5.5.1.1 Splitting the Hamiltonian

Many symplectic discretizations of Hamiltonian dynamics can be derived by "splitting" the Hamiltonian into several terms, and then, for each term in succession, simulating the dynamics defined by that term for some small time step, then repeating this procedure until the desired total simulation time is reached. If the simulation for each term can be done analytically, we obtain a symplectic approximation to the dynamics that is feasible to implement.

This general scheme is described by Leimkuhler and Reich (2004, Section 4.2) and by Sexton and Weingarten (1992). Suppose that the Hamiltonian can be written as a sum of k terms, as follows:

$$H(q,p) = H_1(q,p) + H_2(q,p) + \cdots + H_{k-1}(q,p) + H_k(q,p).$$

Suppose also that we can *exactly* implement Hamiltonian dynamics based on each H_i, for $i = 1, \ldots, k$, with $T_{i,\varepsilon}$ being the mapping defined by applying dynamics based on H_i for time ε. As shown by Leimkuhler and Reich, if the H_i are twice differentiable, the composition of these mappings, $T_{1,\varepsilon} \circ T_{2,\varepsilon} \circ \cdots \circ T_{k-1,\varepsilon} \circ T_{k,\varepsilon}$, is a valid discretization of Hamiltonian dynamics based on H, which will reproduce the exact dynamics in the limit as ε goes to zero, with global error of order ε or less.

Furthermore, this discretization will preserve volume, and will be symplectic, since these properties are satisfied by each of the $T_{i,\varepsilon}$ mappings. The discretization will also be reversible if the sequence of H_i is symmetrical—that is, $H_i(q,p) = H_{k-i+1}(q,p)$. As mentioned at the end of Section 5.2.3, any reversible method must have global error of even order in ε (Leimkuhler and Reich, 2004, Section 4.3.3), which means that the global error must be of order ε^2 or better.

We can derive the leapfrog method from a symmetrical splitting of the Hamiltonian. If $H(q,p) = U(q) + K(p)$, we can write the Hamiltonian as

$$H(q,p) = \frac{U(q)}{2} + K(p) + \frac{U(q)}{2},$$

which corresponds to a split with $H_1(q,p) = H_3(q,p) = U(q)/2$ and $H_2(q,p) = K(p)$. Hamiltonian dynamics based on H_1 is (Equations 5.1 and 5.2):

$$\frac{dq_i}{dt} = \frac{\partial H_1}{\partial p_i} = 0,$$
$$\frac{dp_i}{dt} = -\frac{\partial H_1}{\partial q_i} = -\frac{1}{2}\frac{\partial U}{\partial q_i}.$$

Applying this dynamics for time ε just adds $-(\varepsilon/2)\,\partial U/\partial q_i$ to each p_i, which is the first part of a leapfrog step (Equation 5.18). The dynamics based on H_2 is as follows:

$$\frac{dq_i}{dt} = \frac{\partial H_2}{\partial p_i} = \frac{\partial K}{\partial p_i},$$
$$\frac{dp_i}{dt} = -\frac{\partial H_2}{\partial q_i} = 0.$$

If $K(p) = \frac{1}{2}\sum p_i^2/m_i$, applying this dynamics for time ϵ results in adding $\varepsilon p_i/m_i$ to each q_i, which is the second part of a leapfrog step Equation 5.19. Finally, H_3 produces the third part of a leapfrog step (Equation 5.20), which is the same as the first part, since $H_3 = H_1$.

5.5.1.2 Splitting to Exploit Partial Analytical Solutions

One situation where splitting can help is when the potential energy contains a term that can, on its own, be handled analytically. For example, the potential energy for a Bayesian posterior distribution will be the sum of minus the log prior density for the parameters and minus the log likelihood. If the prior is Gaussian, the log prior density term will be quadratic, and can be handled analytically (see the one-dimensional example at the end of Section 5.2.1).

We can modify the leapfrog method for this situation by using a modified split. Suppose that $U(q) = U_0(q) + U_1(q)$, with U_0 being analytically tractable, in conjunction with the kinetic energy, $K(p)$. We use the split

$$H(q,p) = \frac{U_1(q)}{2} + \big[U_0(q) + K(p)\big] + \frac{U_1(q)}{2}, \tag{5.29}$$

that is, $H_1(q,p) = H_3(q,p) = U_1(q)/2$ and $H_2(q,p) = U_0(q) + K(p)$. The first and last half steps for p are the same as for ordinary leapfrog, based on U_1 alone. The middle full step for q, which in ordinary leapfrog just adds εp to q, is replaced by the analytical solution for following the exact dynamics based on the Hamiltonian $U_0(q) + K(p)$ for time ε.

With this procedure, it should be possible to use a larger stepsize (and hence use fewer steps in a trajectory), since part of the potential energy has been separated out and handled exactly. The benefit of handling the prior exactly may be limited, however, since the prior is usually dominated by the likelihood.

5.5.1.3 Splitting Potential Energies with Variable Computation Costs

Splitting can also help if the potential energy function can be split into two terms, one of which requires less computation time to evaluate than the other (Sexton and Weingarten,

1992). Suppose that $U(q) = U_0(q) + U_1(q)$, with U_0 being cheaper to compute than U_1, and let the kinetic energy be $K(p)$. We can use the following split, for some $M > 1$:

$$H(q,p) = \frac{U_1(q)}{2} + \sum_{m=1}^{M} \left[\frac{U_0(q)}{2M} + \frac{K(p)}{M} + \frac{U_0(q)}{2M} \right] + \frac{U_1(q)}{2}.$$

We label the $k = 3M + 2$ terms as $H_1(q,p) = H_k(q,p) = U_1(q)/2$ and, for $i = 1, \ldots, M$, $H_{3i-1}(q,p) = H_{3i+1}(q,p) = U_0(q)/2M$ and $H_{3i}(q,p) = K(p)/M$. The resulting discretization can be seen as a nested leapfrog method. The M inner leapfrog steps involve only U_0, and use an effective stepsize of ε/M. The outer leapfrog step takes half steps for p using only U_1, and replaces the update for q in the middle with the M inner leapfrog steps.

If U_0 is much cheaper to compute than U_1, we can use a large value for M without increasing computation time by much. The stepsize, ε, that we can use will then be limited mostly by the properties of U_1, since the effective stepsize for U_0 is much smaller, ε/M. Using a bigger ε than with the standard leapfrog method will usually be possible, and hence we will need fewer steps in a trajectory, with fewer computations of U_1.

5.5.1.4 Splitting according to Data Subsets

When sampling from the posterior distribution for a Bayesian model of independent data points, it may be possible to save computation time by splitting the potential energy into terms for subsets of the data.

Suppose that we partition the data into subsets S_m, for $m = 1, \ldots, M$, typically of roughly equal size. We can then write the log likelihood function as $\ell(q) = \sum_{m=1}^{M} \ell_m(q)$, where ℓ_m is the log likelihood function based on the data points in S_m. If $\pi(q)$ is the prior density for the parameters, we can let $U_m(q) = -\log(\pi(q))/M - \ell_m(q)$, and split the Hamiltonian as follows:

$$H(q,p) = \sum_{m=1}^{M} \left[\frac{U_m(q)}{2} + K(p) \Big/ M + \frac{U_m(q)}{2} \right];$$

that is, we let the $k = 3M$ terms be $H_{3m-2}(q,p) = H_{3m}(q,p) = U_m(q)/2$ and $H_{3m-1}(q,p) = K(p)/m$. The resulting discretization with stepsize ε effectively performs M leapfrog steps with stepsize ε/M, with the mth step using MU_m as the potential energy function.

This scheme can be beneficial if the data set is redundant, with many data points that are similar. We then expect $MU_m(q)$ to be approximately the same as $U(q)$, and we might hope that we could set ε to be M times larger than with the standard leapfrog method, obtaining similar results with M times less computation. In practice, however, the error in H at the end of the trajectory will be larger than with standard leapfrog, so the gain will be less than this. I found (Neal, 1996a, Sections 3.5.1 and 3.5.2) that the method can be beneficial for neural network models, especially when combined with the windowed HMC procedure described below in Section 5.5.4.

Note that unlike the other examples above, this split is *not* symmetrical, and hence the resulting discretization is not reversible. However, it can still be used to produce a proposal for HMC as long as the labeling of the subsets is randomized for each iteration, so that the reverse trajectory has the same probability of being produced as the forward trajectory. (It is possible, however, that some symmetrical variation on this split might produce better results.)

5.5.1.5 Handling Constraints

An argument based on splitting shows how to handle constraints on the variables being sampled. Here, I will consider only separate constraints on some subset of the variables, with the constraint on q_i taking the form $q_i \le u_i$, or $q_i \ge l_i$, or both. A similar scheme can handle constraints taking the form $G(q) \ge 0$, for any differentiable function G.

We can impose constraints on variables by letting the potential energy be infinite for values of q that violate any of the constraints, which will give such points probability zero. To see how to handle such infinite potential energies, we can look at a limit of potential energy functions that approach infinity, and the corresponding limit of the dynamics.

To illustrate, suppose that $U_*(q)$ is the potential energy ignoring constraints, and that q_i is constrained to be less than u_i. We can take the limit as $r \to \infty$ of the following potential energy function (which is one of many that could be used):

$$U(q) = U_*(q) + C_r(q_i, u_i), \quad \text{where } C_r(q_i, u_i) = \begin{cases} 0, & \text{if } q_i \le u_i, \\ r^{r+1}(q_i - u_i)^r, & \text{if } q_i > u_i. \end{cases}$$

It is easy to see that $\lim_{r\to\infty} C_r(q_i, u_i)$ is zero for any $q_i \le u_i$ and infinity for any $q_i > u_i$. For any finite $r > 1$, $U(q)$ is differentiable, so we can use it to define Hamiltonian dynamics.

To simulate the dynamics based on this $U(q)$, with a kinetic energy $K(p) = \frac{1}{2}\sum p_i^2/m_i$, we can use the split of Equation 5.29, with $U_1(q) = U_*(q)$ and $U_0(q) = C_r(q_i, u_i)$:

$$H(q,p) = \frac{U_*(q)}{2} + \big[C_r(q_i, u_i) + K(p)\big] + \frac{U_*(q)}{2}.$$

This produces a variation on the leapfrog method in which the half steps for p (Equations 5.18 and 5.19) remain the same, but the full step for q (Equation 5.19) is modified to account for the constraint on q_i. After computing $q_i' = q_i(t) + \varepsilon p_i(t + \varepsilon/2)/m_i$, we check if $q_i' > u_i$. If not, the value of $C_r(q_i, u_i)$ must be zero all along the path from q_i to q_i', and we can set $q(t + \varepsilon)$ to q_i'. But if $q_i' > u_i$, the dynamics based on the Hamiltonian $C_r(q_i, u_i) + K(p)$ will be affected by the C_r term. This term can be seen as a steep hill, which will be climbed as q_i moves past u_i, until the point is reached where C_r is equal to the previous value of $\frac{1}{2}p_i^2/m_i$, at which point p_i will be zero. (If r is sufficiently large, as it will be in the limit as $r \to \infty$, this point will be reached before the end of the full step.) We will then fall down the hill, with p_i taking on increasingly negative values, until we again reach $q_i = u_i$, when p_i will be just the negative of the original value of p_i. We then continue, now moving in the opposite direction, away from the upper limit.

If several variables have constraints, we must follow this procedure for each, and if a variable has both upper and lower constraints, we must repeat the procedure until neither constraint is violated. The end result is that the full step for q of Equation 5.19 is replaced by the procedure shown in Figure 5.8. Intuitively, the trajectory just bounces off the "walls" given by the constraints. If $U_*(q)$ is constant, these bounces are the only interesting aspect of the dynamics, and the procedure is sometimes referred to as "billiards" (see, e.g. Ruján, 1997).

5.5.2 Taking One Step at a Time—The Langevin Method

A special case of HMC arises when the trajectory used to propose a new state consists of only a single leapfrog step. Suppose that we use the kinetic energy $K(p) = \frac{1}{2}\sum p_i^2$. An

```
For each variable, i=1,...,d:

  1) Let p'_i = p_i(t+ε/2)
  2) Let q'_i = q_i(t)+εp'_i/m_i
  3) If q_i is constrained, repeat the following until q'_i satisfies
     all constraints:
      a)  If q_i has an upper constraint, and q'_i > u_i
             Let q'_i=u_i-(q'_i-u_i) and p'_i=-p'_i
      b)  If q_i has a lower constraint, and q'_i < l_i
             Let q'_i=l_i+(l_i-q'_i) and p'_i=-p'_i
  4) Let q_i(t+ε)=q'_i and p_i(t+ε/2)=p'_i
```

FIGURE 5.8
Modification to the leapfrog update of q (Equation 5.19) to handle constraints of the form $q_i \leq u_i$ or $q_i \leq l_i$.

iteration of HMC with one leapfrog step can be expressed in the following way. We sample values for the momentum variables, p, from their Gaussian distributions with mean zero and variance one, and then propose new values, q^* and p^*, as follows:

$$q_i^* = q_i - \frac{\varepsilon^2}{2}\frac{\partial U}{\partial q_i}(q) + \varepsilon p_i, \tag{5.30}$$

$$p_i^* = p_i - \frac{\varepsilon}{2}\frac{\partial U}{\partial q_i}(q) - \frac{\varepsilon}{2}\frac{\partial U}{\partial q_i}(q^*). \tag{5.31}$$

We accept q^* as the new state with probability

$$\min\left[1,\ \exp\left(-(U(q^*) - U(q)) - \frac{1}{2}\sum_i ((p_i^*)^2 - p_i^2)\right)\right], \tag{5.32}$$

and otherwise keep q as the new state. Equation 5.30 is known in physics as one type of "Langevin equation," and this method is therefore known as *Langevin Monte Carlo* (LMC) in the lattice field theory literature (e.g. Kennedy, 1990).

One can also remove any explicit mention of momentum variables, and view this method as performing a Metropolis–Hastings update in which q^* is proposed from a Gaussian distribution where the q_i^* are independent, with means of $q_i - (\varepsilon^2/2)[\partial U/\partial q_i](q)$ and variances of ε^2. Since this proposal is not symmetrical, it must be accepted or rejected based both on the ratio of the probability densities of q^* and q and on the ratio of the probability densities for proposing q from q^* and vice versa (Hastings, 1970). To see the equivalence with HMC using one leapfrog step, we can write the Metropolis–Hastings acceptance probability as follows:

$$\min\left[1,\ \frac{\exp(-U(q^*))}{\exp(-U(q))}\prod_{i=1}^{d}\frac{\exp\left(-\left(q_i - q_i^* + (\varepsilon^2/2)\,[\partial U/\partial q_i](q^*)\right)^2/2\varepsilon^2\right)}{\exp\left(-\left(q_i^* - q_i + (\varepsilon^2/2)\,[\partial U/\partial q_i](q)\right)^2/2\varepsilon^2\right)}\right]. \tag{5.33}$$

To see that this is the same as Equation 5.32, note that using Equations 5.30 and 5.31, we can write

$$p = \frac{1}{\varepsilon}\left[q_i^* - q_i + \frac{\varepsilon^2}{2}\frac{\partial U}{\partial q_i}(q)\right],$$

$$p^* = -\frac{1}{\varepsilon}\left[q_i - q_i^* + \frac{\varepsilon^2}{2}\frac{\partial U}{\partial q_i}(q^*)\right].$$

After substituting these into Equation 5.32, it is straightforward to see the equivalence to Equation 5.33.

In this Metropolis–Hastings form, the LMC method was first proposed by Rossky et al. (1978) for use in physical simulations. Approximate Langevin methods without an accept/reject step can also be used (for a discussion of this, see Neal, 1993, Section 5.3)—as, for instance, in a paper on statistical inference for complex models by Grenander and Miller (1990), where also an accept/reject step is proposed in the discussion by J. Besag (p. 591).

Although LMC can be seen as a special case of HMC, its properties are quite different. Since LMC updates are reversible, and generally make only small changes to the state (since ε typically cannot be very large), LMC will explore the distribution via an inefficient random walk, just like random-walk Metropolis updates.

However, LMC has better scaling behavior than random-walk Metropolis as dimensionality increases, as can be seen from an analysis paralleling that in Section 5.4.4 (Creutz, 1988; Kennedy, 1990). The local error of the leapfrog step is of order ε^3, so $\mathrm{E}[\Delta_1^2]$, the average squared error in H from one variable, will be of order ε^6. From Equation 5.27, $\mathrm{E}[\Delta]$ will also be of order ε^6, and with d independent variables, $\mathrm{E}[\Delta_d]$ will be of order $d\varepsilon^6$, so that ε must scale as $d^{-1/6}$ in order to maintain a reasonable acceptance rate. Since LMC explores the distribution via a random walk, the number of iterations needed to reach a nearly independent point will be proportional to ε^{-2}, which grows as $d^{1/3}$, and the computation time to reach a nearly independent point grows as $d^{4/3}$. This is better than the d^2 growth in computation time for random-walk Metropolis, but worse than the $d^{5/4}$ growth when HMC is used with trajectories that are long enough to reach a nearly independent point.

We can also find what the acceptance rate for LMC will be when the optimal ε is used, when sampling a distribution with independent variables replicated d times. As for random-walk Metropolis and HMC, the acceptance rate is given in terms of $\mu = \mathrm{E}[\Delta_d]$ by Equation 5.28. The cost of obtaining a nearly independent point using LMC is proportional to $1/(a(\mu)\varepsilon^2)$, and since μ is proportional to ε^6, we can write the cost as

$$C_{\mathrm{LMC}} \propto \frac{1}{(a(\mu)\mu^{1/3})}.$$

Numerical calculation shows that this is minimized when $a(\mu)$ is 0.57, a result obtained more formally by Roberts and Rosenthal (1998). This may be useful for tuning, if the behavior of LMC for the distribution being sampled resembles its behavior when sampling for replicated independent variables.

5.5.3 Partial Momentum Refreshment: Another Way to Avoid Random Walks

The single leapfrog step used in the LMC algorithm will usually not be sufficient to move to a nearly independent point, so LMC will explore the distribution via an inefficient random

walk. This is why HMC is typically used with trajectories of many leapfrog steps. An alternative that can suppress random-walk behavior even when trajectories consist of just one leapfrog step is to only partially refresh the momentum between trajectories, as proposed by Horowitz (1991).

Suppose that the kinetic energy has the typical form $K(p) = p^T M^{-1} p/2$. The following update for p will leave invariant the distribution for the momentum (Gaussian with mean zero and covariance M):

$$p' = \alpha p + (1 - \alpha^2)^{1/2} n. \tag{5.34}$$

Here, α is any constant in the interval $[-1, +1]$, and n is a Gaussian random vector with mean zero and covariance matrix M. To see this, note that if p has the required Gaussian distribution, the distribution of p' will also be Gaussian (since it is a linear combination of independent Gaussians), with mean 0 and covariance $\alpha^2 M + (1 - \alpha^2)M = M$.

If α is only slightly less than one, p' will be similar to p, but repeated updates of this sort will eventually produce a value for the momentum variables almost independent of the initial value. When $\alpha = 0$, p' is just set to a random value drawn from its Gaussian distribution, independent of its previous value. Note that when M is diagonal, the update of each momentum variable, p_i, is independent of the updates of other momentum variables.

The partial momentum update of Equation 5.34 can be substituted for the full replacement of the momentum in the standard HMC algorithm. This gives a generalized HMC algorithm in which an iteration consists of three steps:

1. Update the momentum variables using Equation 5.34. Let the new momentum be p'.
2. Propose a new state, (q^*, p^*), by applying L leapfrog steps with stepsize ε, starting at (q, p'), and then negating the momentum. Accept (q^*, p^*) with probability

$$\min\left[1, \exp\left(-U(q^*) + U(q) - K(p^*) + K(p')\right)\right].$$

   If (q^*, p^*) is accepted, let $(q'', p'') = (q^*, p^*)$; otherwise, let $(q'', p'') = (q, p')$.
3. Negate the momentum, so that the new state is $(q'', -p'')$.

The transitions in each of these steps—$(q, p) \to (q, p')$, $(q, p') \to (q'', p'')$, and $(q'', p'') \to (q'', -p'')$—leave the canonical distribution for (q, p) invariant. The entire update therefore also leaves the canonical distribution invariant. The three transitions also each satisfy detailed balance, but the sequential combination of the three does *not* satisfy detailed balance (except when $\alpha = 0$). This is crucial, since if the combination were reversible, it would still result in random-walk behavior when L is small.

Note that omitting step (3) above would result in a valid algorithm, but then, far from suppressing random walks, the method (with α close to one) would produce nearly back-and-forth motion, since the direction of motion would reverse with every trajectory accepted in step (2). With the reversal in step (3), motion continues in the same direction as long as the trajectories in step (2) are accepted, since the two negations of p will cancel. Motion reverses whenever a trajectory is rejected, so if random-walk behavior is to be suppressed, the rejection rate must be kept small.

If $\alpha = 0$, the above algorithm is the same as standard HMC, since step (1) will completely replace the momentum variables, step (2) is the same as for standard HMC, and step (3) will

have no effect, since the momentum will be immediately replaced anyway, in step (1) of the next iteration.

Since this algorithm can be seen as a generalization of standard HMC, with an additional α parameter, one might think it will offer an improvement, provided that α is tuned for best performance. However, Kennedy and Pendleton (2001) show that when the method is applied to high-dimensional multivariate Gaussian distributions only a small constant factor improvement is obtained, with no better scaling with dimensionality. Best performance is obtained using long trajectories (L large), and a value for α that is not very close to one (but not zero, so the optimum choice is not standard HMC). If L is small, the need to keep the rejection rate very low (by using a small ε), as needed to suppress random walks, makes the method less advantageous than standard HMC.

It is disappointing that only a small improvement is obtained with this generalization when sampling a multivariate Gaussian, due to limitations that likely apply to other distributions as well. However, the method may be more useful than one would think from this. For reasons discussed in Sections 5.4.3 and 5.4.5, we will often combine HMC updates with other MCMC updates (perhaps for variables not changed by HMC). There may then be a tradeoff between using long trajectories to make HMC more efficient, and using shorter trajectories so that the other MCMC updates can be done more often. If shorter-than-optimal trajectories are to be used for this reason, setting α greater than zero can reduce the random-walk behavior that would otherwise result.

Furthermore, rejection rates can be reduced using the "window" method described next. An analysis of partial momentum refreshment combined with the window method might find that using trajectories of moderate length in conjunction with a value for α greater than zero produces a more substantial improvement.

5.5.4 Acceptance Using Windows of States

Figure 5.3 (right plot) shows how the error in H varies along a typical trajectory computed with the leapfrog method. Rapid oscillations occur, here with a period of between 2 and 3 leapfrog steps, due to errors in simulating the motion in the most confined direction (or directions, for higher-dimensional distributions). When a long trajectory is used to propose a state for HMC, it is essentially random whether the trajectory ends at a state where the error in H is negative or close to zero, and hence will be accepted with probability close to one, or whether it happens to end at a state with a large positive error in H, and a correspondingly lower acceptance probability. If somehow we could smooth out these oscillations, we might obtain a high probability of acceptance for all trajectories.

I introduced a method for achieving this result that uses "windows" of states at the beginning and end of the trajectory (Neal, 1994). Here, I will present the method as an application of a general technique in which we probabilistically map to a state in a different space, perform a Markov chain transition in this new space, and then probabilistically map back to our original state space (Neal, 2006).

Our original state space consists of pairs, (q, p), of position and momentum variables. We will map to a sequence of W pairs, $[(q_0, p_0), \ldots, (q_{W-1}, p_{W-1})]$, in which each (q_i, p_i) for $i > 0$ is the result of applying one leapfrog step (with some fixed stepsize, ε) to (q_{i-1}, p_{i-1}). Note that even though a point in the new space seems to consist of W times as many numbers as a point in the original space, the real dimensionality of the new space is the same as the old, since the whole sequence of W pairs is determined by (q_0, p_0).

To probabilistically map from (q, p) to a sequence of pairs, $[(q_0, p_0), \ldots, (q_{W-1}, p_{W-1})]$, we select s uniformly from $\{0, \ldots, W-1\}$, and set (q_s, p_s) in the new state to our current state

(q, p). The other (q_i, p_i) pairs in the new state are obtained using leapfrog steps from (q_s, p_s), for $i > s$, or backwards leapfrog steps (i.e. done with stepsize $-\varepsilon$) for $i < s$. It is easy to see, using the fact that leapfrog steps preserve volume, that if our original state is distributed with probability density $P(q, p)$, then the probability density of obtaining the sequence $[(q_0, p_0), \ldots, (q_{W-1}, p_{W-1})]$ by this procedure is

$$P\left([(q_0, p_0), \ldots, (q_{W-1}, p_{W-1})]\right) = \frac{1}{W} \sum_{i=0}^{W-1} P(q_i, p_i), \tag{5.35}$$

since we can obtain this sequence from a (q, p) pair that matches any pair in the sequence, and the probability is $1/W$ that we will produce the sequence starting from each of these pairs (which happens only if the random selection of s puts the pair at the right place in the sequence).

Having mapped to a sequence of W pairs, we now perform a Metropolis update that keeps the sequence distribution defined by Equation 5.35 invariant, before mapping back to the original state space. To obtain a Metropolis proposal, we perform $L - W + 1$ leapfrog steps (for some $L \geq W-1$), starting from (q_{W-1}, p_{W-1}), producing pairs (q_W, p_W) to (q_L, p_L). We then propose the sequence $[(q_L, -p_L), \ldots, (q_{L-W+1}, -p_{L-W+1})]$. We accept or reject this proposed sequence by the usual Metropolis criterion, with the acceptance probability being

$$\min \left[1, \frac{\sum_{i=L-W+1}^{L} P(q_i, p_i)}{\sum_{i=0}^{W-1} P(q_i, p_i)} \right], \tag{5.36}$$

with $P(q, p) \propto \exp(-H(q, p))$. (Note here that $H(q, p) = H(q, -p)$, and that starting from the proposed sequence would lead symmetrically to the original sequence being proposed.)

This Metropolis update leaves us with either the sequence $[(q_L, p_L), \ldots, (q_{L-W+1}, p_{L-W+1})]$, called the "accept window," or the sequence $[(q_0, p_0), \ldots, (q_{W-1}, p_{W-1})]$, called the "reject window." (Note that these windows will overlap if $L + 1 < 2W$.) We label the pairs in the window chosen as $[(q_0^+, p_0^+), \ldots, (q_{W-1}^+, p_{W-1}^+)]$. We now produce a final state for the windowed HMC update by probabilistically mapping from this sequence to a single pair, choosing (q_e^+, p_e^+) with probability

$$\frac{P(q_e^+, p_e^+)}{\sum_{i=0}^{W-1} P(q_i^+, p_i^+)}.$$

If the sequence in the chosen window was distributed according to Equation 5.35, the pair (q_e^+, p_e^+) chosen will be distributed according to $P(q, p) \propto \exp(-H(q, p))$, as desired. To see this, let (q_{e+n}^+, p_{e+n}^+) be the result of applying n leapfrog steps (backward ones if $n < 0$) starting at (q_e^+, p_e^+). The probability density that (q_e^+, p_e^+) will result from mapping from a sequence to a single pair can then be written as follows, considering all sequences that can contain (q_e^+, p_e^+) and their probabilities:

$$\sum_{k=e-W+1}^{e} \left[\frac{1}{W} \sum_{i=k}^{k+W-1} P(q_i^+, p_i^+) \right] \frac{P(q_e^+, p_e^+)}{\sum_{i=k}^{k+W-1} P(q_i^+, p_i^+)} = P(q_e^+, p_e^+).$$

The entire procedure therefore leaves the correct distribution invariant.

When $W > 1$, the potential problem with ergodicity discussed at the end of Section 5.3.2 does not arise, since there is a nonzero probability of moving to a state only one leapfrog step away, where q may differ arbitrarily from its value at the current state.

It might appear that the windowed HMC procedure requires saving all $2W$ states in the accept and reject windows, since any one of these states might become the new state when a state is selected from either the accept window or reject window. Actually, however, at most three states need to be saved—the start state, so that forward simulation can be resumed after the initial backward simulation, plus one state from the reject window and one state from the accept window, one of which will become the new state after one of these windows is chosen. As states in each window are produced in sequence, a decision is made whether the state just produced should replace the state presently saved for that window. Suppose that the sum of the probability densities of states seen so far is $s_i = p_1 + \cdots + p_i$. If the state just produced has probability density p_{i+1}, it replaces the previous state saved from this window with probability $p_{i+1}/(s_i + p_{i+1})$.

I showed (Neal, 1994) that, compared to standard HMC, using windows improves the performance of HMC by a factor of 2 or more, on multivariate Gaussian distributions in which the standard deviation in some directions is much larger than in other directions. This is because the acceptance probability in Equation 5.36 uses an average of probability densities over states in a window, smoothing out the oscillations in H from inexact simulation of the trajectory. Empirically, the advantage of the windowed method was found to increase with dimensionality. For high-dimensional distributions, the acceptance probability when using the optimal stepsize was approximately 0.85, larger than the theoretical value of 0.65 for HMC (see Section 5.4.4).

These results for multivariate Gaussian distributions were obtained with a window size, W, much less than the trajectory length, L. For less regular distributions, it may be advantageous to use a much larger window. When $W = L/2$, the acceptance test determines whether the new state is from the first half of the trajectory (which includes the current state) or the second half; the new state is then chosen from one half or the other with probabilities proportional to the probability densities of the states in that half. This choice of W guards against the last few states of the trajectory having low probability density (high H), as might happen if the trajectory had by then entered a region where the stepsize used was too big.

The windowed variant of HMC may make other variants of HMC more attractive. One such variant (Section 5.5.1) splits the Hamiltonian into many terms corresponding to subsets of the data, which tends to make errors in H higher (while saving computation). Errors in H have less effect when averaged over windows. As discussed in Section 5.5.3, very low rejection rates are desirable when using partial momentum refreshment. It is easier to obtain a low rejection probability using windows (i.e. a less drastic reduction in ϵ is needed), which makes partial momentum refreshment more attractive.

Qin and Liu (2001) introduced a variant on windowed HMC. In their version, L leapfrog steps are done from the start state, with the accept window consisting of the states after the last W of these steps. A state from the accept window is then selected with probabilities proportional to their probability densities. If the state selected is k states before the end, k backwards leapfrog steps are done from the start state, and the states found by these steps along with those up to $W - k - 1$ steps forward of the start state form the reject window. The state selected from the accept window then becomes the next state with probability given by the analog of Equation 5.36; otherwise the state remains the same.

Qin and Liu's procedure is quite similar to the original windowed HMC procedure. One disadvantage of Qin and Liu's procedure is that the state is unchanged when the accept window is rejected, whereas in the original procedure a state is selected from the reject window (which might be the current state, but often will not be). The only other difference is that the number of steps from the current state to an accepted state ranges from $L - W + 1$ to L (average $L - (W + 1)/2$) with Qin and Liu's procedure, versus from $L - 2W + 2$

to L (average $L - W + 1$) for the original windowed HMC procedure, while the number of leapfrog steps computed varies from L to $L + W - 1$ with Qin and Liu's procedure, and is fixed at L with the original procedure. These differences are slight if $W \ll L$. Qin and Lin claim that their procedure performs better than the original on high-dimensional multivariate Gaussian distributions, but their experiments are flawed.*

Qin and Liu (2001) also introduce the more useful idea of weighting the states in the accept and reject windows nonuniformly, which can be incorporated into the original procedure as well. When mapping from the current state to a sequence of W weighted states, the position of the current state is chosen with probabilities equal to the weights, and when computing the acceptance probability or choosing a state from the accept or reject window, the probability densities of states are multiplied by their weights. Qin and Liu use weights that favor states more distant from the current state, which could be useful by usually causing movement to a distant point, while allowing choice of a nearer point if the distant points have low probability density. Alternatively, if one sees a window as a way of smoothing the errors in H, symmetrical weights that implement a better "low pass filter" would make sense.

5.5.5 Using Approximations to Compute the Trajectory

The validity of HMC does not depend on using the correct Hamiltonian when simulating the trajectory. We can instead use some approximate Hamiltonian, as long as we simulate the dynamics based on it by a method that is reversible and volume-preserving. However, the exact Hamiltonian must be used when computing the probability of accepting the endpoint of the trajectory. There is no need to look for an approximation to the kinetic energy, when it is of a simple form such as Equation 5.13, but the potential energy is often much more complex and costly to compute—for instance, it may involve the sum of log likelihoods based on many data points, if the data cannot be summarized by a simple sufficient statistic. When using trajectories of many leapfrog steps, we can therefore save much computation time if a fast and accurate approximation to the potential energy is available, while still obtaining exact results (apart from the usual sampling variation inherent in MCMC).

Many ways of approximating the potential energy might be useful. For example, if its evaluation requires iterative numerical methods, fewer iterations might be done than are necessary to get a result accurate to machine precision. In a Bayesian statistical application, a less costly approximation to the unnormalized posterior density (whose log gives the potential energy) may be obtainable by simply looking at only a subset of the data. This may not be a good strategy in general, but I have found it useful for Gaussian process models (Neal, 1998; Rasmussen and Williams, 2006), for which computation time scales as the cube of the number of data points, so that even a small reduction in the number of points produces a useful speedup.

Rasmussen (2003) has proposed approximating the potential energy by modeling it as a Gaussian process, inferred from values of the potential energy at positions selected during an initial exploratory phase. This method assumes only a degree of smoothness of the potential energy function, and so could be widely applied. It is limited, however, by the cost of

* In their first comparison, their method computes an average of 55 leapfrog steps per iteration, but the original only computes 50 steps, a difference in computation time which if properly accounted for negates the slight advantage they see for their procedure. Their second comparison has a similar problem, and it is also clear from an examination of the results (in their Table I) that the sampling errors in their comparison are too large for any meaningful conclusions to be drawn.