Markov Chain Monte Carlo Algorithm Comparisons

by

Sijin Wen

M.A., York University 1998

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Master of Science
in
THE FACULTY OF GRADUATE STUDIES
(Department of Statistics)

we accept this thesis as conforming
to the required standard

The University of British Columbia
April 2001
© Sijin Wen, 2001
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Statistics
The University of British Columbia
Vancouver, Canada

Date April 19, 2001
Abstract

Various Markov chain Monte Carlo algorithms are available for sampling from a posterior distribution. The random walk Metropolis algorithm is a simple scheme which is frequently used in Bayesian statistical problem. The guided walk algorithm attempts to suppress the random walk behavior in the random walk Metropolis algorithm. Other algorithms, such as the Langevin algorithm and the hybrid algorithm use more information about the posterior distribution than the random walk Metropolis algorithm and the guided walk algorithm. In this thesis, The performance of each of those four algorithms has been examined, based on simulation studies using multivariate normal target distributions. Then we compare the algorithms in terms of efficiency and convergence time. Moreover, these four algorithms are compared using a posterior distribution for parameters given observed data in an application.
Contents

Abstract ii

Contents iii

List of Tables v

List of Figures vii

Acknowledgements x

Dedication xi

1 Introduction 1

2 Background on general state-space Markov chain theory 5

2.1 Notation and definition 6

2.2 Irreducibility and recurrence 9

2.3 Convergence 11

3 Markov Chain Monte Carlo Algorithms 13
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Summary of Markov chain Monte Carlo algorithms</td>
<td>25</td>
</tr>
<tr>
<td>4.1</td>
<td>Simulation results with 'one-at-a-time' parameters updating, based on a 5-variate normal distribution</td>
<td>34</td>
</tr>
<tr>
<td>4.2</td>
<td>Simulation results with 'all-at-a-time' parameters updating, based on a 5-variate normal distribution</td>
<td>35</td>
</tr>
<tr>
<td>4.3</td>
<td>Simulation results with 'one-at-a-time' parameters updating, based on a 20-variate normal distribution</td>
<td>36</td>
</tr>
<tr>
<td>4.4</td>
<td>Simulation results with 'all-at-a-time' parameters updating, based on a 20-variate normal distribution</td>
<td>37</td>
</tr>
<tr>
<td>4.5</td>
<td>Mean number of iteration to convergence, based on a 5-variate Normal Distribution</td>
<td>38</td>
</tr>
<tr>
<td>4.6</td>
<td>Mean number of iteration to convergence, based on a 20-variate Normal Distribution</td>
<td>39</td>
</tr>
<tr>
<td>5.1</td>
<td>Iterations to convergence with one-at-a-time updating</td>
<td>54</td>
</tr>
<tr>
<td>5.2</td>
<td>Iterations to Convergence with all-at-a-time updating</td>
<td>55</td>
</tr>
</tbody>
</table>
5.3 Estimation of the regression parameter with the Monte Carlo standard error (one-at-a-time updating): 57

5.4 Estimation of the regression parameter with the Monte Carlo standard error (all-at-a-time updating): 57
List of Figures

3.1 Sample paths of the RW, GW, Langevin and hybrid algorithms for a standard normal target distribution. The acceptance rate is 0.45 for the GW algorithm, 0.9 for the GW algorithm, 0.99 both for the Langevin and hybrid algorithm. The initial state is $x_0 = 5$ in each case. ........................................ 28

4.1 Comparisons of algorithms with one-at-a-time updating, based on a 5-variate normal distribution. The upper panel compares the mean FIT statistics for the four algorithm. The other four panels contain boxplots for the $m = 200$ FIT statistics obtained using the RW, GW, Langevin and hybrid algorithm, respectively, as a function of acceptance rate. ......................... 42
4.2 Comparisons of algorithms with one-at-a-time updating, based on a 20-variate normal distribution. The upper panel compares the mean FIT statistics for the four algorithm. The other four panels contain boxplots for the $m = 200$ FIT statistics obtained using the RW, GW, Langevin and hybrid algorithm, respectively, as a function of acceptance rate.  

4.3 Comparisons of algorithms with all-at-a-time updating, based on a 5-variate normal distribution. The upper panel compares the mean FIT statistics for the four algorithm. The other four panels contain boxplots for the $m = 200$ FIT statistics obtained using the RW, GW, Langevin and hybrid algorithm, respectively, as a function of acceptance rate.  

4.4 Comparisons of algorithms with all-at-a-time updating, based on a 20-variate normal distribution. The upper panel compares the mean FIT statistics for the four algorithm. The other four panels contain boxplots for the $m = 200$ FIT statistics obtained using the RW, GW, Langevin and hybrid algorithm, respectively, as a function of acceptance rate.  

4.5 Comparisons of the speed of the convergence between the algorithms. Each panel contains boxplots of four algorithms with 200 of the same starting points from the uniform distribution on the hyper-square $(0, 20) \times \ldots \times (0, 20)$.  

5.1 Histograms of the three variables of the tree data
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>Comparison of the speed of convergence for one-at-a-time updating, based on how fast the shrink factor for $\alpha$ approaches one</td>
<td>60</td>
</tr>
<tr>
<td>5.3</td>
<td>Comparison of the speed of convergence for one-at-a-time updating, based on how fast the shrink factor for $\beta_1$ approaches one</td>
<td>61</td>
</tr>
<tr>
<td>5.4</td>
<td>Comparison of the speed of convergence for one-at-a-time updating, based on how fast the shrink factor for $\beta_2$ approaches one</td>
<td>62</td>
</tr>
<tr>
<td>5.5</td>
<td>Comparison of the speed of convergence for all-at-a-time updating, based on how fast the shrink factor for $\alpha$ approaches one</td>
<td>63</td>
</tr>
<tr>
<td>5.6</td>
<td>Comparison of the speed of convergence for all-at-a-time updating, based on how fast the shrink factor $\beta_1$ approaches one</td>
<td>64</td>
</tr>
<tr>
<td>5.7</td>
<td>Comparison of the speed of convergence for all-at-a-time updating, based on how fast the shrink factor $\beta_2$ approaches one</td>
<td>65</td>
</tr>
</tbody>
</table>
Acknowledgements

Foremost I would like to thank my thesis supervisor, Dr. Paul Gustafson, for his guidance in the development of my thesis, for his excellent advice and support and for his great patience. Without him, this thesis would never have been completed. I am very grateful to Dr. Lang Wu for his careful reading of the manuscript, his valuable comments and suggestions. Furthermore, I would like to thank for all support and help from all the faculty members, staff and graduate students in the department.

Finally, I would like to thank my wife, Panqing Wu, for her continued patience and support.

Sijin Wen

The University of British Columbia
April 2001
To my mother, and to my daughter
Chapter 1

Introduction

In a surprisingly short period, Markov chain Monte Carlo (MCMC) techniques have emerged as extremely popular tools for analysis of complex statistical models. This is especially true in the field of Bayesian analysis. In particular, MCMC algorithms allow a user to draw inferences from a complex posterior distribution on a high-dimensional parameter space, by simulating a Markov chain which has the posterior distribution as its stationary distribution. Under weak conditions the chain converges to its stationary distribution, so that posterior quantities can be estimated from the simulation output. On the other hand, in many such settings, alternative methodologies such as asymptotic approximation, traditional numerical quadrature, and non-iterative Monte Carlo methods either might be infeasible or fail to provide sufficiently accurate results. Important features of MCMC methods that enhance their applicability include their ability to reduce complex multidimensional problems to a sequence of much lower-dimensional ones and their relative indifference to the
presence or absence of conjugate structure between the likelihood and the prior
distribution (Cowles and Carlin, 1996).

Many MCMC algorithms are available for sampling from a posterior dis­
tribution. The Gibbs sampler (Geman and Geman, 1984; Gelfand and Smith,
1990) requires draws from the posterior conditional distribution of each pa­
rameter, given the other parameters. In general, the Gibbs sampler is more
efficient if the dimension of the posterior distribution is low, or the full con­
dditional distributions are standard distributions. The biggest disadvantage of
this method is that sampling from complex multivariate distributions might
be not possible when the full conditional distributions are not standard dis­
tributions. Moreover, its computational inefficiency might increase with the
dimensionality of the posterior distribution.

The random walk Metropolis algorithm (Metropolis et al., 1953; Hast­
ings, 1970) operates by proposing that the Markov chain move to a candidate
state obtained by adding noise to the current state. This algorithm is com­
monly used in practice since it is easy to implement. Meanwhile, it exhibits
some random walk behavior in its exploration of the posterior distribution.
That is, the direction in which the Markov chain attempts to move is ran­
domized at each transition and therefore the algorithm seems inefficient. The
guided walk algorithm (Gustafson, 1998) attempts to suppress the random
walk behavior to some extent, allowing the chain to move more quickly through
the posterior distribution. In particular, the direction of the candidate is not
randomized at each transition, and the chain moves consistently in the same
direction until a candidate state is rejected.

In general, one expects algorithms which use more information to be more efficient. The random walk Metropolis and guided walk algorithms require only unnormalized posterior density evaluation. However, the Langevin algorithm (Roberts and Rosenthal, 1998) and the hybrid algorithm (Duane et al., 1987; Neal, 1993; Gustafson, 1997 and 1998) use more information about the posterior distribution. They require evaluations of the log unnormalized posterior density, along with its first partial derivatives. Note also that the hybrid algorithm can be viewed as modified the Langevin algorithm with random walk suppression.

What we are interested in is as follows. When faced with a model fitting problem, we must decide whether an algorithm that uses log-posterior derivatives will be used; whether the parameters will be updated 'one-at-a-time' (i.e., with component by component updating) or 'all-at-time' (i.e., with all components updating); and whether an algorithm that attempts to suppress random walks will be used. The various algorithms have not yet been compared carefully. In this thesis, the random walk Metropolis algorithm, guided walk algorithm, Langevin algorithm and hybrid algorithm will be compared.

The algorithm comparisons will be based mainly on simulation studies using multivariate normal target distributions. First, a $\chi^2$ goodness of fit statistic is used to measure how well a simulated chain represents the target distribution. Then the simulation studies compare how many iterations each algorithm requires to reach the stationary distribution, using the same starting
points for each algorithm chain. On the other hand, we are also interested in some applied work involving MCMC methods. Comparing with the simulation study, the stationary distribution in practice is not known in closed-form, and this will complicate our comparisons between algorithms. For example, at what point is it reasonable to believe that the samples are truly representative of the underlying stationary distribution of the Markov chain? Consequently, MCMC convergence diagnostics will be much more difficult than in the simulation study. The convergence diagnostics of Gelman and Rubin (1992) are popular in the statistical community. This method then is used in our application, where a posterior distribution of parameters given observed data is examined using each of those four MCMC algorithms.

The thesis is organized as follows. Chapter 2 contains background on general state-space Markov chain theory. Chapter 3 includes introductions to the four MCMC algorithms: the random walk Metropolis algorithm, the guided walk Metropolis algorithm, the Langevin algorithm and the hybrid algorithm. These four MCMC algorithms are compared in Chapter 4, based on simulation studies using multivariate normal target distributions. Chapter 5 compares the algorithms using a posterior distribution for parameters given observed data. We conclude with a discussion in Chapter 6.
Chapter 2

Background on general state-space Markov chain theory

Markov chain Monte Carlo is a method for exploring a distribution \( \pi \) in which a Markov chain with invariant distribution \( \pi \) is constructed and sample path averages of this Markov chain are used to estimate characteristics of \( \pi \). The distribution \( \pi \) is assumed to be defined on a set \( E \) called the state space. Often \( E \) will be \( n \)-dimensional Euclidean space \( \mathbb{R}^n \), but it can be more general. The technical requirement is that the collection \( \mathcal{E} \) of subsets of \( E \) on which \( \pi \) is defined must be a countably generated \( \sigma \)-algebra.

This chapter summarizes some results of general-state space Markov chain theory, which are required in developing Monte Carlo method on Markov chains. All results given in this chapter are derived from results presented in Nummelin (1984) and Meyn and Tweedie (1993). Many of these results are given in Tierney (1994) with proofs.
2.1 Notation and definition

Definition Let $\mathcal{E}$ be a countably generated $\sigma$-algebra on $E$. A Markov transition kernel on $(E, \mathcal{E})$ is a map $P : E \times \mathcal{E} \to [0, 1]$ such that:

- for any fixed $A \in \mathcal{E}$, the function $P(., A)$ is measurable;
- for any fixed $x \in E$, the function $P(x, .)$ is a probability measure on $(E, \mathcal{E})$.

For example, the transition kernel of a time-homogeneous Markov chain $\{X_n\} = \{X_n, n = 0, 1, \ldots\}$ on the state-space $E$ is a function $P(x, A)$ such that for any $n \geq 0$

$$P\{X_{n+1} \in A | X_n = x\} = P(x, A)$$

for all $x \in E$ and $A \subset E$. That is, $P(x, .)$ is the distribution of Markov chain after one step given that it starts at $x$. Different strategies, such as the Gibbs sampler, Metropolis-Hastings algorithms, etc., give rise to different transition kernels. The distribution of a time-homogeneous Markov chain $\{X_n\}$ is specified by its initial distribution and its transition kernel. Note that transition kernels are the general state-space versions of discrete chain transition matrices.

For simplicity, all subsets of $E$ and all functions defined on $E$ used in this chapter will be assumed to be $\mathcal{E}$-measurable, and $\pi$ will be assumed to have a density, which will be denoted by $\pi(x)$.
Definition For a probability distribution $\nu$ and a real-valued, $\mathcal{E}$-measurable function $h$, the expectation of $h$ is denoted by

$$\nu h = \int h(x)\nu(dx),$$

where $\nu(dx)$ is the probability under $\nu$ of a small measurable subset $dx \in \mathcal{E}$; the integral is over such subsets $dx$.

For example, if $\nu$ is discrete with probability mass function $f(x)$, then

$$\nu h = \sum h(x)f(x),$$

and if $\nu$ is continuous with density $f(x)$, then

$$\nu h = \int h(x)f(x)dx.$$

Definition For a probability distribution $\nu$ and a transition kernel $P$ on $(\mathcal{E}, \mathcal{E})$, define the distribution $\nu P$ by

$$(\nu P)(A) = \int P(x,A)\nu(dx),$$

where $\nu(dx)$ is the probability under $\nu$ of a small measurable subset $dx \in \mathcal{E}$; the integral is over such subsets $dx$. That is, the distribution $\nu P$ is the distribution of position of a Markov chain with transition kernel $P$ and initial distribution $\nu$ after one step.

Definition A Markov chain has invariant (stationary) distribution $\pi$ if $\pi = \pi P$.

Note that an invariant distribution $\pi$ over the states of a Markov chain is one that persists forever once it is reached.
**Definition** For a real-valued, $\mathcal{E}$-measurable function $h$ and a transition function $P$ on $(E, \mathcal{E})$, the function $Ph$ is defined as

$$(Ph)(x) = \int h(y)P(x, dy) = E[h(X_1)|X_0 = x],$$

where $P(x, dy)$ is the probability of moving to a small measurable subset $dy \in E$ given that the move states at $x$; and the integral is over such subsets $dy$.

**Definition** The product $PQ$ of two transition kernels $P$ and $Q$ is the transition kernel defined by

$$PQ(x, A) = \int Q(y, A)P(x, dy)$$

for all $x \in E$ and $A \subset E$, where $P(x, dy)$ is the probability of moving to a small measurable subset $dy \in E$ given that the move states at $x$; and the integral is over such subsets $dy$.

The $n^{th}$ iterate $P^n$ of $P$ is defined recursively for $n \geq 1$ by $P^1 = P$ and $P^n = PP^{n-1}$ for $n \geq 2$. Let $P^0$ be the identity kernel that puts probability one on remaining at the initial value, then we can write

$$P\{X_n \in A|X_0 = x\} = P^n(x, A)$$

for all $n \geq 0$.

**Definition** A probability distribution $\nu_1$ is **absolutely continuous** with respect to a probability distribution $\nu_2$ if every null set of $\nu_2$ is a null set of $\nu_1$, i.e., if $\nu_2(A) = 0$ implies $\nu_1(A) = 0$ for any $A \subset E$. Two probability distributions are said to be equivalent if they have the same null sets.
**Definition** The variation distance between two probability distributions $\nu_1$ and $\nu_2$ is defined as

$$||\nu_1 - \nu_2|| = 2 \sup_{A \in E} |\nu_1(A) - \nu_2(A)|.$$ 

Note that for a probability distribution $\nu$ on $E$ a statement holds for $\nu$-almost all $x$ if $\nu$ gives probability zero to the set of points in $E$ where the statement fails.

**Notation**

- Probabilities and expectations for a Markov chain started with $X_0 = x$ are denoted by $P_x$ and $E_x$, respectively. Probabilities and expectations for a Markov chain with initial distribution $\nu$ are denoted by $P_\nu$ and $E_\nu$.

- The first return time of Markov chain to a set $A \subseteq E$ is denoted by $\tau_A$. 
  $\tau_A = \infty$ if the chain never returns to $A$.

### 2.2 Irreducibility and recurrence

**Definition** A Markov chain is $\phi$-irreducible for a probability distribution $\phi$ on $E$ if $\phi(A) > 0$ for a set $A \subseteq E$ implies that

$$P_x\{\tau_A < \infty\} > 0$$

for all $x \in E$. A chain is irreducible if it is $\phi$-irreducible for some probability distribution $\phi$. If a chain is $\phi$-irreducible, then $\phi$ is called an irreducibility distribution for the chain.

**Notes**
• If a chain is irreducible, then it might have many different irreducibility distributions.

• Any irreducible chain has a maximal irreducibility distribution \( \psi \) in sense that all other irreducibility distributions are absolutely continuous with respect to \( \psi \). Maximal irreducibility distributions are not unique but are equivalent, i.e., they have the same null sets.

**Definition** An irreducible Markov chain with maximal irreducibility distribution \( \psi \) is **recurrent** if, for any sets \( A \subseteq E \) with \( \psi(A) > 0 \),

- \( P_x \{ X_n \in A \text{ infinitely often} \} > 0 \) for all \( x \),
- \( P_x \{ X_n \in A \text{ infinitely often} \} = 1 \) for \( \psi \)-almost all \( x \).

Moreover, the chain is **Harris recurrent** if \( P_x \{ X_n \in A \text{ infinitely often} \} = 1 \) for all \( x \).

**Definition** An irreducible recurrent chain is **positive recurrent** if it has an invariant probability distribution. Otherwise it is null recurrent.

Irreducibility means that all interesting sets can be reached. Recurrence is the property that all such sets will be reached infinitely often, at least from almost all starting points.

**Theorem** Suppose the Markov chain \( \{ X_n \} \) is irreducible and has invariant distribution \( \pi \). Then the chain is \( \pi \)-irreducible, \( \pi \) is a maximal irreducibility distribution, \( \pi \) is the unique invariant distribution of the chain, and the chain is positive recurrent.
It should be noted that in MCMC, we already have a target distribution \( \pi \) so that by the above theorem, \( \pi \) is the unique invariant distribution of the chain, and the chain is positive recurrent if we can demonstrate irreducibility. Moreover, irreducibility is a crucial concept in the convergence theory of general state chain is that the Markov chain.

### 2.3 Convergence

**Definition** An \( m \)-cycle for an irreducible chain with transition kernel \( P \) is a collection \( \{E_0, \ldots, E_{m-1}\} \) of disjoint sets such that \( P(x, E_j) = 1 \) for \( j = i + 1 \pmod{m} \) and all \( x \in E_i \). The period \( d \) of the chain is the largest \( m \) for which an \( m \)-cycle exists. The chain is *aperiodic* if \( d = 1 \).

**Fundamental Theorem** Suppose \( \{X_n\} \) is an irreducible Markov chain with transition kernel \( P \) and invariant distribution \( \pi \). Define the average transition kernel \( \tilde{P}^n \) by

\[
\tilde{P}^n(x, A) = \frac{1}{n+1} \sum_{i=0}^{n} P^i(x, A)
\]

for all \( x \in E \) and \( A \subset E \). Then

\[
||\tilde{P}^n(x, \cdot) - \pi(\cdot)|| \to 0
\]

for \( \pi \)-almost all \( x \) (for all \( x \) if the chain is Harris recurrent). Moreover, if the chain is aperiodic, then

\[
||P^n(x, \cdot) - \pi(\cdot)|| \to 0
\]

for \( \pi \)-almost all \( x \) (for all \( x \) if the chain is Harris recurrent).
It turns out that, for the Markov chain to converge to an invariant distribution, the chain needs to satisfy three important properties. First, it has to be irreducible. That is, from all starting points, the chain can reach any non-empty set with positive probability, in some number of iterations. Second, the chain must be positive recurrent. This can be expressed in terms of the existence of a stationary distribution such that if the initial value is sampled from the stationary distribution, then all subsequent iterates will also distributed according to the stationary distribution. Finally, the chain needs to be aperiodic. This stops the Markov chain from oscillating between different sets of states in a regular periodic movement.

Finally, suppose that we have a Markov chain with transition kernel $P$ which has invariant distribution $\pi$. For a MCMC experiment using this chain to be successful, we need to ensure that the sample path average

$$\bar{f}_n = \frac{1}{n+1} \sum_{i=0}^{n} f(X_i)$$

converges to the expectation $\pi f$ for any initial distribution whenever this expectation exists. A strong law of large numbers can be described as follows.

**Theorem** Suppose $\{X_n\}$ is an irreducible Markov chain with transition kernel $P$ and invariant distribution $\pi$, and let $f$ be a real-valued function on $E$ such that $\pi|f| < \infty$. Then $P_x\{\bar{f}_n \to \pi f\} = 1$ for $\pi$-almost all $x$ (for all $x$ if the chain is Harris recurrent).
Chapter 3

Markov Chain Monte Carlo Algorithms

Many approaches are available for constructing Markov chains with a specified invariant distribution. Four MCMC algorithms, including the random walk Metropolis, guided walk Metropolis, Langevin and hybrid algorithms, will be introduced in this chapter. Meanwhile, some similarities and differences between these four algorithms will be discussed during the introduction of the algorithms.

3.1 The Metropolis - Hastings algorithm

The Metropolis-Hastings algorithm is the fundamental building block of most MCMC algorithms. We describe the form due to Hastings (1970), which is a generalization of the method originally introduced by Metropolis et al. (1953).
To define Hastings’ version of the algorithm, suppose that $\pi$ has a density with respect to $\mu$. We wish to construct a Markov chain $X^1, X^2, \ldots, X^t, \ldots$ with $\pi$ as its equilibrium distribution. Let $Q$ be a Markov transition kernel of the form

$$Q(x, dy) = q(x, y)\mu(dy).$$

If $X^t = x$ is the current state of the chain, then the Metropolis-Hastings algorithm proceeds by simulating a candidate or proposal value $y$ from the transition density, $q(x, \cdot)$. The next state, $X^{t+1}$, is then randomly assigned to be either $y$ with probability $\alpha(x, y)$, or $x$ with probability $1 - \alpha(x, y)$, where

$$\alpha(x, y) = \min\left\{\frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}, 1\right\}$$

is the acceptance probability.

Define the density of a Metropolis transitional kernel as

$$p(x, y) = \begin{cases} q(x, y)\alpha(x, y), & \text{if } x \neq y, \\ 0, & \text{if } x = y, \end{cases}$$

and set

$$r(x) = 1 - \int p(x, y)\mu(dy)$$

for the probability that the algorithm remains at $x$. Thus the Metropolis kernel $P$ can be written as

$$P(x, dy) = p(x, y)\mu(dy) + r(x)\delta_x(dy),$$

where $\delta_x$ denotes point mass at $x$.

Note that $p$ satisfies the reversibility (also known as detailed balance) condition

$$\pi(x)p(x, y) = \pi(y)p(y, x).$$
In fact, it is trivial when \( x = y \); when \( x \neq y \),

\[
\pi(x)p(x, y) = \pi(x)q(x, y)\alpha(x, y) = \pi(x)q(x, y)\min\left\{ \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}, 1 \right\} = \min\left\{ \pi(y)q(y, x), \pi(x)q(x, y) \right\} = \pi(y)q(y, x)\min\left\{ 1, \frac{\pi(x)q(x, y)}{\pi(y)q(y, x)} \right\} = \pi(y)p(y, x),
\]

it follows that \( \pi \) is an invariant distribution for \( P \): for any measurable set \( A \),

\[
\int P(x, A)\pi(dx) = \int \left\{ \int p(x, y)\mu(dy) \right\} \pi(x)\mu(dx) + \int r(x)\delta_x(A)\pi(x)\mu(dx) = \int \left\{ \int p(x, y)\pi(x)\mu(dx) \right\} \mu(dy) + \int r(x)\pi(x}\mu(dx) = \int \left\{ \int p(y, x)\pi(y)\mu(dx) \right\} \mu(dy) + \int r(x)\pi(x)\mu(dx) = \int (1 - r(y))\pi(y)\mu(dy) + \int r(x)\pi(x)\mu(dx) = \int \pi(y)\mu(dy),
\]

that is, \( \pi P = \pi \).

The algorithm is very general, allowing a variety of useful special cases based on different choices of the transition densities \( q \). In the case that \( \pi \) is a continuous distribution on \( \mathbb{R}^n \), the random walk Metropolis algorithm (RW) works as follows. The candidate state is obtained by adding noise to the current state; specifically, \( q(x, y) = f(y - x) \), for some density \( f \) which is some pre-defined spherically-symmetric proposal distribution. Commonly \( f \) is taken to be the normal density with zero mean vector and covariance matrix \( \epsilon^2 I \), where \( I \) is the \( n \times n \) identity matrix. In general, one expects that if the noise
is small (a small value of \( \epsilon \) is chosen), then rejections should be infrequent. The algorithm for updating from \( X^t = x \) to the next realized state \( X^{t+1} = x' \) can be expressed as:

\[
x^* \leftarrow x + \epsilon p, \quad \text{where} \quad p \sim N(0, I)
\]

\[
\alpha \leftarrow \min\left\{ \frac{\pi(x^*)}{\pi(x)}, 1 \right\}
\]

\[
x' \leftarrow \begin{cases} 
    x^* \quad \text{with probability } \alpha, \\
    x \quad \text{with probability } 1 - \alpha.
\end{cases}
\]

If the density \( f \) is strictly positive (for example, a normal density) on all of \( \mathbb{R}^n \), then the random walk Metropolis kernel is irreducible and aperiodic (Tierney, 1994). Moreover, the random walk Metropolis kernel is Harris recurrent (Tierney, 1994).

Note that this algorithm only depends on \( \pi \) through ratios of the form \( \pi(y)/\pi(x) \); thus \( \pi \) only needs to be known up to a normalizing constant. Also, the symmetry property of the proposal transition, \( q(x, y) = q(y, x) \), leads to a simple form for the acceptance probability.

Note also that the Gibbs sampler can be viewed as a special case of the Metropolis-Hasting algorithm where the proposal transition density is chosen according to full conditional densities. Let \( \pi(x) = \pi(x_1, \ldots, x_n) \) denote the joint density, and let \( \pi(x_i|x_{-i}) \) denote the induced full conditional densities for each of the components \( x_i \), given values of the other components \( x_{-i} = (x_j; j \neq i) \). Then the Gibbs sampler requires successive generation from the conditional forms \( \pi(x_i|x_{-i}) \) by regarding \( \pi(x) \) as a function of \( x_i \) only, considering \( x_j \),
\( j \neq i \), to be fixed. For example, the candidate value \( x^* \) is simulated from \((x_1, ...x_{i-1}, z, x_{i+1}, ..., x_n)\), where \( z \) is generated from \( \pi(x_i|x_{-i}) \). This implies that the acceptance probability for \( x^* \) will be
\[
\alpha = \min \left\{ \frac{\pi(x^*)\pi(x_i|x_{-i})}{\pi(x)\pi(z|x_{-i})}, 1 \right\}
\]
\[
= \min \left\{ \frac{\pi(z|x_{-i})\pi(x_{-i})\pi(x_i|x_{-i})}{\pi(x_i|x_{-i})\pi(x_{-i})\pi(z|x_{-i})}, 1 \right\}
\]
\[
= 1.
\]

3.2 The guided walk algorithm

At each transition of the random walk algorithm, each component of the candidate state is generated from a symmetric distribution centered at the current state. To suppress this random walk behavior, we now propose a guided walk Metropolis algorithm as follows. Extend the state from \( X \) to \((X, P)\), to yield a target distribution with density:
\[
\pi(x, p) = \pi(x)e^{-||p||^2}
\]
where \( p \) is a vector of latent variables with \( \dim(x) = \dim(p) \) and \( ||p||^2 = \sum_i p_i^2 \). Thus under the target distribution the components of \( p \) have independent standard normal distributions. Clearly, we can use a simulated chain with this extended stationary distribution to learn about \( \pi(x) \), simply by discarding the sampled \( p \) states. A transition of the Guided Walk (GW) algorithm for updating from \((X^t, P^t) = (x, p)\) to \((X^{t+1}, P^{t+1}) = (x', p')\) can be expressed as:
\[
x^* \leftarrow x + cp, \text{ where } p \sim N(0, I)
\]
\[ \alpha \leftarrow \min \left\{ \frac{\pi(x^*)}{\pi(x)}, 1 \right\} \]

\[ (x', p') \leftarrow \begin{cases} (x^*, p) & \text{with probability } \alpha, \\ (x, -p) & \text{with probability } 1 - \alpha. \end{cases} \]

\[ p' \leftarrow \rho p' + (1 - \rho^2)^{1/2} z \]

where \( z \sim N(0, I) \) and \( \rho \in [0, 1) \). A large value of \( \rho \) is needed to suppress random walk behavior.

To see that the GW transition leaves the extended target distribution invariant, note that it can be expressed as the composition of three transitions, each of which individually leaves the extended target invariant. Starting from \((x, p)\), the first transition is a Metropolis-Hastings update with candidate state \((x + \epsilon p, -p)\). Under this transition, a move from the current state to the candidate state is equally as likely as the time-reversed move from the candidate state to the current state. This yields a simple form for the acceptance probability. The second transition is simply a negation of the latent vector \( p \). This is easily verified to leave \( \pi(x, p) \) invariant, due to the symmetry of normal distribution. The third is an autoregressive transition on \( p \) only:

\[ p^{(new)} = \rho p^{(old)} + (1 - \rho^2)^{1/2} z, \]

which is a vector with independently distributed standard normal components since \( p^{(old)} \) and \( z \) are, thereby leaving \( \pi(x, p) \) invariant. Note that the guided algorithm is an adaptation of an idea of Horowitz (1991) from the more sophisticated hybrid MCMC setting as in Section 3.3. Also, Gustafson (1998) considered a guided walk algorithm with univariate parameter updating to
suppress the random walk behavior, where the latent variable was a discrete distribution satisfying $pr(P = 1) = pr(P = -1) = 0.5$. There is a similar idea in the algorithm we propose. In fact, this algorithm works by implementing a univariate GW step in a particular direction and then slightly perturbing the direction vector in a manner which leaves the extended target distribution invariant.

In contrast to the RW algorithm, in the GW algorithm the direction of the candidate $x^*$ relative to current state $x$ is not randomized at each transition. When $\rho$ is chosen to be close to 1, the chain is likely to move consistently in the same direction until a candidate state is rejected. Upon rejection, the chain begins a series of moves in the opposite direction. Note that when $\rho$ is close to 0, this algorithm will be equivalent to RW algorithm. Note also that if a small value of $\epsilon$ is chosen, then rejections, and consequently reversals, should be infrequent.

### 3.3 The hybrid Monte Carlo algorithm:

The variant of the hybrid algorithm due to Horowitz (1991) is discussed by Neal (1993). This method involves construction of a Markov chain on the enlarged parameter space of $(x, p)$, to yield a stationary distribution with density:

$$
\pi(x, p) \propto \exp\{-E(x) + K(p)\}
$$

where $E(x)$ is the negative log posterior density, and $p$ is a vector of latent variables with $\dim(x) = \dim(p) = k$, $K(p) = (1/2) \sum_{i=1}^{k} p_i^2$, implying that the
components of \( p \) have independent standard normal distributions. Then, the \( x \) marginal of a simulated chain is treated as a dependent sample from the posterior.

The method is better motivated using the physical analogy in which the algorithm is rooted. In particular, \( x \) and \( p \) can be regarded as position and momentum vectors for a physical system, with \( E(x) \) and \( K(p) \) being potential and kinetic energy respectively. The combination of position and momentum variables is known as phase space. So the total energy function for points in phase space, known as the Hamiltonian, is \( H(x,p) = E(x) + K(p) \). The distribution (1) on \( (x,p) \) is then the Boltzmann or steady-state distribution over the phase space.

Then, according to the Hamiltonian formulation of Newtonian dynamics, we have the differential equations

\[
\frac{dx}{dt} = p \\
\frac{dp}{dt} = -\nabla E(x)
\]

where \( \nabla E(x) \) is the gradient of \( E \) evaluated at \( x \). In practice, the dynamics must be followed along a discretized time grid, which is known as the leapfrog discretization:

\[
x(t + \epsilon) = x(t) + \epsilon \{p(t) - (\epsilon/2)\nabla E(x(t))\},
\]

\[
p(t + \epsilon) = p(t) - (\epsilon/2)\{\nabla E(x(t)) + \nabla E(x(t + \epsilon))\}.
\]

The leapfrog discretization has the important property of time reversibility. That is, two successive applications of the transition correspond to an
identity transition in which \((x, p)\) is unchanged. Another important property of the leapfrog discretization is that it preserves phase space volume since each increment of \(x\) or \(p\) depends only on a value of the other's, producing a 'shear' transformation that does not change a region's volume (see Neal (1993) for further discussion of this point). When these are applied iteratively, there is a simpler form for updating, where the updates for \(p\) leapfrog in time over those for \(x\), and vice versa. See Neal (1993) for a detailed discussion of the discretization.

The algorithm for updating from \((X^t, P^t) = (x, p)\) to \((X^{t+1}, P^{t+1}) = (x', p')\) proceeds as

\[
x^* \leftarrow x + \epsilon \{ p + (\epsilon/2) \nabla \log \pi(x) \}
\]

\[
p^* \leftarrow -p - (\epsilon/2) \{ \nabla \log \pi(x) + \nabla \log \pi(x^*) \}
\]

\[
\alpha \leftarrow \min \left\{ \frac{\pi(x^*, p^*)}{\pi(x, p)}, 1 \right\}
\]

\[
(x', p') \leftarrow \begin{cases} 
(x^*, p^*) & \text{with probability } \alpha, \\
(x, p) & \text{with probability } 1 - \alpha.
\end{cases}
\]

\[
p' \leftarrow -p'
\]

\[
p' \leftarrow \rho p' + (1 - \rho^2)^{1/2} z
\]

where \(z \sim N(0, I)\) and \(\rho \in [0, 1)\). A large value of \(\rho\) is needed to suppress random walk behavior.
The complete hybrid transition of chain is given by the composition of three transitions, each of which leaves \( \pi(x, p) \) as the stationary distribution. The most important and involved of the three transitions is the first, which is based on a Metropolis-Hastings step. A candidate state \((x^*, p^*)\) is obtained from the current state \((x, p)\) according to (2) and (3). This is motivated as a discretized approximation to the evolution of a dynamical system over a time period of \( \epsilon \), followed by negation of the momentum vector. If there were no approximation due to discretization, then such an evolution would be along a path of total energy, given \( H(x^*, p^*) = H(x, p) \). If \( \epsilon \) is not too large, then the fluctuation in \( H \) due to discretization will be small. This makes it likely that the candidate step will be accepted. The other important property of the dynamical evolution is time reversibility. The important consequence of this for chain evolution is that transition from \((x, p)\) to \((x^*, p^*)\) is just as likely as transition from \((x^*, p^*)\) to \((x, p)\). Consequently, no adjustment factor is required in the Metropolis-Hastings acceptance probability. Thus with probability \( \alpha \) the new state is set equal to the candidate state \((x^*, p^*)\), and with probability \( 1 - \alpha \) the new state is set equal to the old state \((x, p)\).

The second transition is simply a negation of the momentum vector \( p \), which leaves \( \pi(x, p) \) as the stationary distribution due to the symmetry of the normal distribution. The combined effect of transitions 1 and 2 is that the new \( p \) state is the same as the old \( p \) state, provide there was as acceptance in the Metropolis-Hastings step. As a result, if there are few rejections, then the dynamics will be free to evolve without frequent momentum negations.
Consequently, more of the state space can be explored in fewer iterations.

The only changes in $H$ associated with the above transitions result from discretization error. Therefore a third transition is required to ensure that the chain visits states of varying total energy $H(x, p)$. It is simplest to accomplish this via transitions which operate on $p$ only. One possibility is to sample a new value of $p$ from its marginal distribution (independent standard normal components). However, this interferes with the dynamical evolution, and introduces unnecessary random walk behavior into the sample. Alternatively, an autoregressive transition is used

$$p^{\text{new}} = \rho p^{\text{old}} + (1 - \rho^2)^{1/2} z,$$

which is a vector with independently distributed standard normal components since $p^{\text{old}}$ and $z$ are, therefore leaves $\pi(x, p)$ invariant.

Both the GW and hybrid transitions can be expressed as the composition of three transitions, each of which individually leaves the extended target invariant. In contrast to the RW algorithm, the direction of the candidate $x^*$ relative to current state $x$ is not randomized at each transition in both of the GW and hybrid algorithms. The main difference between the hybrid Monte Carlo algorithm and the other two algorithms (the GW and RW algorithms) is that, when a candidate $(x^*, p^*)$ is generated via discretized time evolution, evaluations of the log posterior density along with its first partial derivatives are needed in the hybrid algorithm. Note that the hybrid Monte Carlo algorithm will be reduced to the GW algorithm if evaluations of the log posterior density along with its first partial derivatives are ignored.
3.4 The Langevin Monte Carlo algorithm:

As in the hybrid Monte Carlo method; the Langevin MC algorithm also in­
volves construction of a Markov chain on the enlarged parameter space of
\((x, p)\), to yield a stationary distribution with density (1) in Section 3.3. In
the Langevin Monte Carlo method, after picking a new value \(p\) of the latent
variable \(P\), one can choose to select a candidate state by only a single leapfrog
iteration, accepting or rejecting the resulting state based on the change in the
total energy \(H(x, p) = E(x) + K(p)\). In fact, the algorithm for updating from
\(X^t = x\) to \(X^{t+1} = x'\) proceeds as follows

\[
x^* \leftarrow x + \epsilon \{p + (\epsilon/2) \nabla \log \pi(x)\}, \text{where } p \sim N(0, I)
\]

\[
p^* \leftarrow -p - (\epsilon/2) \left\{ \nabla \log \pi(x) + \nabla \log \pi(x^*) \right\}
\]

\[
\alpha \leftarrow \min \left\{ \frac{\pi(x^*, p^*)}{\pi(x, p^*)}, 1 \right\}
\]

\[
x' \leftarrow \begin{cases} x^* \text{ with probability } \alpha, \\ x \text{ with probability } 1 - \alpha. \end{cases}
\]

This algorithm is the same as the hybrid Monte Carlo algorithm when
\(\epsilon\) is taken to be zero in Section 3.2. As is the case with the hybrid Monte
Carlo method, the Langevin Monte Carlo transition leaves the extended target
distribution invariant. Note that, like the hybrid Monte Carlo method, the
Langevin Monte Carlo method is 'dynamical' in the sense that it makes use of
the derivative of the energy function \(H(t)\). On the other hand, the Langevin
Monte Carlo method has the disadvantage that, unlike the hybrid Monte Carlo
method, it must explore the state space via a random walk.

24
Table 3.1: Summary of Markov chain Monte Carlo algorithms

<table>
<thead>
<tr>
<th>Algorithm Behavior</th>
<th>Derivatives Of Posterior</th>
</tr>
</thead>
<tbody>
<tr>
<td>random walk</td>
<td>not required</td>
</tr>
<tr>
<td>random walk suppression</td>
<td>required</td>
</tr>
<tr>
<td>LW</td>
<td>RW</td>
</tr>
<tr>
<td>LW</td>
<td>GW</td>
</tr>
<tr>
<td>LW</td>
<td>Langevin hybrid</td>
</tr>
</tbody>
</table>

3.5 Remarks

In summary, the random walk Metropolis-Hastings and the Langevin Monte Carlo algorithms exhibit the random walk behavior in their exploration of posterior distribution. This behavior seems inefficient, since it can take many iterations for the chain to traverse a substantial distance, while the hybrid/guided walk algorithm can be viewed as modified Langevin/random walk Metropolis-Hastings algorithm with random walk suppression. On the other hand, the random walk Metropolis-Hastings and guided walk algorithms only involve log posterior density evaluations, whereas the Langevin and hybrid algorithms also incorporate gradient information about the posterior distribution in the sense that they require evaluations of the log unnormalized posterior density along with the first partial derivatives. In general, one expects algorithms which use more information to be more efficient. The trade-off is that more computations are needed. The summary of the differences and similarities between these four algorithms is also given in Table 3.1. Sample paths of the RW, GW, Langevin and hybrid algorithms for a standard normal target distribution are given in Figure 3.1.

Moreover, the following algorithm is useful to implement the above four
MCMC methods on a computer. Suppose that the Markov chain is constructed to have an equilibrium distribution with density

$$
\pi(x, p) \propto \exp\{-E(x) + K(p)\}
$$

where $E(x) = -\log \pi(x)$ is the negative log posterior density, $p$ is a vector of latent variables with $\text{dim}(x) = \text{dim}(p)$, $K(p) = (1/2) \sum_{i} p_i^2$, implying that the components of $p$ have independent standard normal distributions. Then these four algorithms for updating from $(X^t, P^t) = (x, p)$ to $(X^{t+1}, P^{t+1}) = (x', p')$ proceeds as

$$
\begin{align*}
x^* &\leftarrow x + \epsilon \{p + (\epsilon/2)g(x)\} \\
p^* &\leftarrow -p - (\epsilon/2)\{g(x) + g(x^*)\} \\
\alpha &\leftarrow \min \left\{ \frac{\pi(x^*, p^*)}{\pi(x, p)}, 1 \right\} \\
(x', p') &\begin{cases} (x^*, p^*) &\text{with probability } \alpha, \\
(x, p) &\text{with probability } 1 - \alpha.
\end{cases}
\end{align*}
$$

$p' \leftarrow -p'$

$$
p' \leftarrow \rho p' + (1 - \rho^2)^{1/2} z \text{ where } z \sim N(0, I)
$$

- when $g(x) = 0$ and $\rho = 0$: random walk Metropolis algorithm;
- when $g(x) = 0$ and $\rho \approx 1$: guided walk Metropolis algorithm;
- when $g(x) = \nabla \log \pi(x)$ and $\rho = 0$: Langevin algorithm;
- when $g(x) = \nabla \log \pi(x)$ and $\rho \approx 1$: hybrid algorithm.
Note that when we construct a Markov chain $X^1, X^2, ..., X^t, ...$ with a target distribution on $\mathbb{R}^n$, the MCMC algorithm for generating $X^t = (x'_1, ..., x'_n)$ from $X^{t-1} = (x_1, ..., x_n)$ can be expressed by two different procedures:

- **one-at-a-time updating (with univariate updating):**
  
  Updating $x'_1$ for the first component of $X^t$ given $x_2, x_3, ..., x_n$.

  Updating $x'_2$ for the second component of $X^t$ given $x'_1, x_3, ..., x_n$.

  ..........

  Updating $x'_n$ for the last component of $X^t$ given $x'_1, x'_2, ..., x'_{n-1}$.

  Note that the new value of a component of $X^t$ is used immediately when updating the next component of $X^t$.

- **all-at-a-time updating (with multivariate updating):**

  Updating $(x'_1, x'_2, ..., x'_n)$ for the all components of $X^t$ given $(x_1, x_2, ..., x_n)$.

Hence, given a MCMC algorithm, one might be interested in the differences of the algorithm performance between 'one-at-a-time' and 'all-at-a-time' updating. This also is one of our objectives in the thesis. In the following chapter, the RW, GW, Langevin and hybrid algorithms will be compared, based on simulation studies using multivariate normal target distributions.
Figure 3.1: Sample paths of the RW, GW, Langevin and hybrid algorithms for a standard normal target distribution. The acceptance rate is 0.45 for the GW algorithm, 0.9 for the GW algorithm, 0.99 both for the Langevin and hybrid algorithm. The initial state is $x_0 = 5$ in each case.
Chapter 4

Simulation study

4.1 Experimental design

In a Monte Carlo study there are usually several different things (factors) that we want to investigate. As in other kinds of experiments, a factorial design is usually more efficient. Each factor occurs at different levels, and the set of all levels of all factors that are used in the study constitute the 'design space'. At this point, there are three factors of interest in our Monte Carlo experiment, and each factor has two levels:

- For the comparison between the random walk suppression and no suppression, we use: (1) the guided walk algorithm versus the random walk Metropolis algorithm; (2) the hybrid algorithm versus the Langevin algorithm.

- For the parameters updating, we use two methods: (1) univariate updat-
ing (i.e., ‘one-at-a-time’); (2) multivariate updating (i.e., ‘all-at-a-time’).

- For the comparison between using derivatives and no derivatives in the
  log posterior gradient evaluations, we choose: (1) the hybrid MCMC
  algorithm versus the guided walk algorithm, (2) the Langevin algorithm
  versus the random walk Metropolis algorithm.

So this is a $2 \times 2 \times 2$ factorial Monte Carlo experiment.

4.2 Multivariate normal distribution and a $\chi^2$

goodness of fit statistic

The simulation study will compare various MCMC algorithms, based on a
multivariate target distribution. Let the target distribution $\pi$ be $k$-variate
normal, with zero mean vector, and covariance matrix

$$\Sigma = (1 - \rho)I_k + \rho J_k,$$

where $I_k$ is the $k \times k$ identity matrix, and $J_k$ is the $k \times k$ matrix where
every entry is one. Thus $\rho$ is the correlation between any two distinct com-
ponents of $X$. When $\rho$ is close to one, this target distribution will be chal-
lenging for MCMC schemes using component by component updating, since
the conditional distribution of $X_i$ given the other components will be very
narrow compared to the marginal distribution of $X_i$. In particular, let $X_{(i)} =
(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_k)$, and let $\bar{X}_{(i)}$ be the mean of $X_{(i)}$. Then the condi-
tional distribution of $X_i|X(i)$ is normal, with mean

$$E(X_i|X(i)) = \frac{(k - 1)\rho}{1 + (k - 2)\rho} \bar{X}(i)$$

and variance $\text{Var}(X_i|X(i)) = \tau^2$, where

$$\tau^2 = 1 - \rho^2 \left[ \frac{k - 1}{1 + (k - 2)\rho} \right].$$

The partition of the state space used to assess fit is based on transforming the target distribution to the standard multivariate normal distribution (mean vector $0$ and covariance matrix $I_k$). Specifically, if $\mathbf{X} = (X_1, ..., X_k)'$ are distributed according to the target distribution with pdf

$$f(\mathbf{x}) = (2\pi)^{-k/2} |\Sigma|^{-1/2} \exp\{-\mathbf{x}'\Sigma^{-1}\mathbf{x}/2\},$$

then $\mathbf{X}^* = (X_1^*, ..., X_k^*)'$ are standard multivariate normal with pdf

$$f(\mathbf{x}^*) = (2\pi)^{-k/2} \exp\{-x_1^{*2} + ... + x_k^{*2}/2\},$$

where

$$x_i^* = \frac{1}{\sqrt{1 - \rho}} \left\{ x_i - \left( 1 - \sqrt{\frac{1 - \rho}{1 + (k - 1)\rho}} \bar{x} \right) \right\}.$$  

For the algorithms which require evaluations of the log unnormalized posterior density, along with its first partial derivatives, the formula for the first partial derivatives is given by

$$\frac{\partial \log f}{\partial x_i} = -\sum_{i=1}^{k} x_i^* \frac{\partial x_i^*}{\partial x_i}$$

$$= \frac{-1}{\sqrt{1 - \rho}} \left\{ x_i^* - \left( 1 - \sqrt{\frac{1 - \rho}{1 + (k - 1)\rho}} \right) \bar{x}^* \right\}.$$
After transformation, a ‘dart board’ partition is applied to \( X^* \). This partition comprises the \( r = L \times 2^k \) sets obtained by first partitioning the sampled \( X^* \) vectors into \( L \) equiprobable sets on the basis of the magnitude of \( \|X^*\|^2 = (X^*_1)^2 + \ldots + (X^*_k)^2 \) compared to appropriate quantiles of the \( \chi^2 \) distribution. Each of the \( L \) sets is then further partitioned into \( 2^k \) sets on the basis of the pattern of signs of components of \( X^* \).

We use a \( \chi^2 \) goodness of fit statistic which measures how well a simulated chain represents the target distribution, given that the chain is started in equilibrium:

1. **partition the state space into** \( r \) disjoint subsets \( A_1, \ldots, A_r \), with \( \pi(A_i) = 1/r \).

2. \( E = n/r \) where \( n \) is the number of iterations comprising the chain, and \( O_i \) be the count of how many of the \( n \) sampled states lie in \( A_i \).

3. Let

\[
FIT = \left\{ \frac{\sum_{i=1}^{r} (O_i - E)^2}{E} \right\}^{1/2},
\]

then, \( FIT^2 \) would approximately have a \( \chi^2 \) distribution with \( r - 1 \) degrees of freedom, if the sampled values were independent draws from the target distribution.

4. We also consider the root mean square relative error

\[
RMSRE = \left\{ \frac{1}{r} \sum_{i=1}^{r} \left( \frac{O_i - E}{E} \right)^2 \right\}^{1/2},
\]

The \( RMSRE \) can be interpreted as the typical relative error incurred when using a sample proportion to estimate the target probability of a parti-
4.3 Simulations

The first reported simulations are based on a $k = 5$ dimensional target distribution, with correlations $\rho = 0.95$. A collection of $m = 200$ starting vectors is simulated from the target distribution. For each starting point, and each in an arithmetic series of proposal variances $\epsilon$'s, the RW, GW, Langevin and hybrid chains of length $n = 2000$ are simulated. To assess fit a partition of $r = 160$ sets is constructed, using $L = 5$.

As alluded to previously in Chapter 3, the choice of $\epsilon$ will affect the equilibrium acceptance rate. For each algorithm, the arithmetic series of $\epsilon$'s is chosen so that a series of acceptance rates ranging from 20% to 95% is obtained. The best value of $\epsilon$'s is obtained when the mean $FIT$ statistic is smallest. The corresponding equilibrium acceptance rate will be the best acceptance rate for this algorithm. The simulation results, given in Table 4.1 and Table 4.2, are as follows.

- In the situation of 'one-at-a-time', the summary of $FIT$ statistic is shown in Figure 4.1. The mean of $FIT$ statistic is smallest when the acceptance rate is 50% for the RW algorithm, 60% for the GW algorithm, 68% for the Langevin algorithm, and 87% for the hybrid algorithm. Using these respective acceptance rates, we can examine the ratios of mean $FIT$ statistics. Over the set of $m = 200$ starting values, the ratios of mean $FIT$ statistics are that 0.89 for GW to RW, 0.72 for Langevin to
Table 4.1: Simulation results with 'one-at-a-time' parameters updating, based on a 5-variate normal distribution.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best acpt. rate (e)</th>
<th>Mean FIT statistic</th>
<th>Rate of FIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random walk</td>
<td>0.50 (0.48)</td>
<td>21.6</td>
<td>0.89</td>
</tr>
<tr>
<td>Guided walk</td>
<td>0.60 (0.41)</td>
<td>19.3</td>
<td>0.72</td>
</tr>
<tr>
<td>Langevin</td>
<td>0.70 (0.36)</td>
<td>14.0</td>
<td>0.87</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.80 (0.29)</td>
<td>12.2</td>
<td>NA</td>
</tr>
</tbody>
</table>

GW, and 0.87 for hybrid to Langevin. Given the interpretation of FIT statistic, the ratios of mean FIT statistics suggest that relative error can be reduced by 11% as a result of using the GW algorithm instead of the RW algorithm, by 28% using the Langevin algorithm instead of the GW algorithm, and by 13% using the hybrid algorithm instead of the Langevin algorithm. These results imply that the random walk behaviors are suppressed by the GW algorithm and the hybrid algorithm; and the Langevin and hybrid algorithms with information about derivatives of posterior are more efficient than the other two algorithms.

- In the situation of 'all-at-a-time', the summary of FIT statistic is shown in Figure 4.3. The mean of FIT statistic is smallest when the acceptance rate is 38% for the RW algorithm, 51% for the GW algorithm, 80% for the Langevin algorithm, and 87% for the hybrid algorithm. Using these respective acceptance rates, we can examine the ratios of FIT statistics. Over the set of \( m = 200 \) starting values, the ratios of FIT statistics are that 1.04 for GW to RW, 0.63 for Langevin to GW, and 0.7 for hybrid to Langevin. The results are similar as in the case of 'one-at-a-time',

34
Table 4.2: Simulation results with 'all-at-a-time' parameters updating, based on a 5-variate normal distribution.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best acpt. rate (e)</th>
<th>Mean FIT statistic</th>
<th>Rate of FIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random walk</td>
<td>0.40 (0.20)</td>
<td>36.6</td>
<td>1.09</td>
</tr>
<tr>
<td>Guided walk</td>
<td>0.50 (0.16)</td>
<td>39.8</td>
<td>0.58</td>
</tr>
<tr>
<td>Langevin</td>
<td>0.75 (0.25)</td>
<td>23.3</td>
<td>0.69</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.85 (0.17)</td>
<td>16.1</td>
<td>NA</td>
</tr>
</tbody>
</table>

except for the guided walk algorithm. The guided walk algorithm did not suppress the random walk behavior as in the case of 'one-at-a-time'. Our interpretation of the GW algorithm is that, in the situation of 'one-at-a-time', the chain moves consistently in the same direction until a candidate state is rejected. Upon rejection, the chain begins a series of moves in opposite direction in the sense that this component of the latent vector $P$ is negated and the other components have no changes; while in the situation of 'all-at-a-time', there are many directions (more than two, say) based on the signs of the components of $P$. Upon rejection, the chain begins a series of moves in opposite direction in the sense that all components of the latent vector $P$ are negated. Although this direction is slightly perturbed, it is not as meaningful as in the situation of 'one-at-a-time'. However, it should be noted that the improvement can be made with the information of log-posterior derivatives used along, as the hybrid algorithm appears to offer obvious gains over the Langevin algorithm.

- The algorithm with univariate updating (one-at-a-time) appears to offer
Table 4.3: Simulation results with 'one-at-a-time' parameters updating, based on a 20-variate normal distribution.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best acpt. rate (c)</th>
<th>Mean FIT statistic</th>
<th>Rate of FIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random walk</td>
<td>0.55 (0.34)</td>
<td>17.2</td>
<td>0.94</td>
</tr>
<tr>
<td>Guided walk</td>
<td>0.55 (0.34)</td>
<td>16.2</td>
<td>0.79</td>
</tr>
<tr>
<td>Langevin</td>
<td>0.70 (0.37)</td>
<td>12.7</td>
<td>0.81</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.80 (0.31)</td>
<td>11.7</td>
<td>NA</td>
</tr>
</tbody>
</table>

consistent large and obvious gains over those with multivariate updating (all-at-a-time). The ratios of mean $FIT$ statistics for 'one-at-a-time' to 'all-at-a-time' are 0.58 for RW, 0.48 for GW, 0.6 for Langevin and 0.76 for hybrid. As a result of using the algorithms with 'one-at-a-time' instead of those with 'all-at-a-time', the relative error can be reduced by 42% for RW, 52% for GW, 40% for the Langevin algorithm and 24% for the hybrid algorithm.

The second reported simulations are based on a $k = 20$ dimensional target distribution, with the same correlations $\rho = 0.95$ and the same collection of $m = 200$ starting vectors from the target distribution. The first five components of $x$ are used to assess fit. That is, a partition of $r = 160$ sets is constructed, using $L = 5$. Each simulated chain is of length $n = 2000$. The summaries of the $FIT$ statistics are shown in Figure 4.2 and Figure 4.4. The simulation results, given in Table 4.3 and Table 4.4, are similar as in the case of the dimension $k = 5$. 
Table 4.4: Simulation results with ‘all-at-a-time’ parameters updating, based on a 20-variate normal distribution.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best acpt. rate (ε)</th>
<th>Mean FIT statistic</th>
<th>Rate of FIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random walk</td>
<td>0.35 (0.1)</td>
<td>38.1</td>
<td>1.10</td>
</tr>
<tr>
<td>Guided walk</td>
<td>0.35 (0.1)</td>
<td>47.4</td>
<td>0.45</td>
</tr>
<tr>
<td>Langevin</td>
<td>0.80 (0.16)</td>
<td>20.2</td>
<td>0.81</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.85 (0.16)</td>
<td>16.6</td>
<td>NA</td>
</tr>
</tbody>
</table>


4.4 Speed of Convergence

Thus far we have compared these four algorithms in terms of how well their output represents the target distribution, given that the chain is started in equilibrium. In practical problems it is not possible to start the chain in equilibrium, so the speed at which the chain reaches equilibrium is an important issue. We provide the following examples comparing how many iterations each algorithm requires to reach the stationary distribution.

First, we consider the exchangeable multivariate normal target distribution ($k = 5, \rho = 0.95$) as in Section 4.2. We generate $m = 200$ initial state vectors from the uniform distribution on the ‘hyper-square’ $(0, 20) \times \ldots \times (0, 20)$. (Since the target distribution is centered at the origin, we purposefully sample initial vectors from a distribution not centered at the origin, in order to make it more challenging for the algorithms to converge.) For each four algorithms chains are simulated for each starting value, using acceptance rates that were optimal in Section 4.3. Since $X_T \Sigma^{-1} X$ is $\chi^2_k$ distributed under the target distribution, each chain is labeled as having converged once $\frac{x_n^T \Sigma^{-1} x_n}{n}$ is smaller than the 0.95 quantile of the $\chi^2_k$ distribution. A boxplot of the sample sizes
Table 4.5: Mean number of iteration to convergence, based on a 5-variate Normal Distribution.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>one-at-a-time updating</th>
<th>all-at-a-time updating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random walk</td>
<td>274</td>
<td>834</td>
</tr>
<tr>
<td>Guided walk</td>
<td>181</td>
<td>1253</td>
</tr>
<tr>
<td>Langevin</td>
<td>65</td>
<td>245</td>
</tr>
<tr>
<td>Hybrid</td>
<td>68</td>
<td>85</td>
</tr>
</tbody>
</table>

required for convergence of the four algorithms is shown in Figure 4.5. The summary of mean number of iteration to convergence is also given in Table 4.5.

In the case of 'one-at-a-time', over the set of $m = 200$ starting values, the mean number of iterations is 273 for the RW algorithm, 181 for the GW algorithm, 65 for the Langevin algorithm and 68 for the hybrid algorithm. These results imply that the GW algorithm converges more quickly than the RW algorithm, and the Langevin algorithm converges more quickly than the GW algorithm, but there are no big differences between the Langevin and the hybrid algorithm.

In the case of 'all-at-a-time', over the set of $m = 200$ starting values, the mean number of iterations for the guided walk algorithm, given in Table 3, is 1253, which is the biggest number in Table 3. This also shows that the performance of the guided walk algorithm in the case of 'all-at-a-time' is not as good as that in the case of 'one-at-a-time'. The simulation results imply that the RW algorithm converges more quickly than the GW algorithm, the Langevin algorithm converges more quickly than the RW algorithm, and the
Table 4.6: Mean number of iteration to convergence, based on a 20-variate Normal Distribution.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>one-at-a-time updating</th>
<th>all-at-a-time updating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random walk</td>
<td>766</td>
<td>9510</td>
</tr>
<tr>
<td>Guided walk</td>
<td>558</td>
<td>24442</td>
</tr>
<tr>
<td>Langevin</td>
<td>213</td>
<td>1225</td>
</tr>
<tr>
<td>Hybrid</td>
<td>145</td>
<td>294</td>
</tr>
</tbody>
</table>

hybrid algorithm converges more quickly than the Langevin algorithm.

Note also that all four algorithms with 'one-at-a-time' updating converge more quickly than those with 'all-at-a-time' updating.

Another scenario for comparing the four algorithms involves a higher dimensional exchangeable multivariate normal distribution (k = 20) with the same correlations ρ = 0.95 and the same collection of m = 200 starting vectors from the uniform distribution on the same 'hyper-square' (0, 20) × ... × (0, 20). The simulation results, given in Table 4.6, are similar as in the case of the dimension k = 5.

4.5 Concluding remarks

• The comparison between using derivatives and no derivatives in the log posterior gradient evaluations

The algorithms using log-posterior derivatives appear to offer consistent gains over those not using log-posterior derivatives in the simulation study. For example, when dimension k = 20 the algorithms using log-
posterior derivatives offer a mean reduction in relative error on the order of 50\% with multivariate updating, at least 20\% with univariate updating. In addition, the algorithms using log-posterior derivatives tend to converge in at least one-eighth fewer iterations for multivariate updating, and at least half fewer iterations for univariate updating. There is a similar result when dimension $k = 5$.

- **The comparison between the random walk suppression and no suppression**

With univariate updating, the efficiency and convergence time gains are achieved by random walk suppression with the GW or hybrid algorithm. Note that with multivariate updating, the performance of the GW algorithm is not as good as the RW algorithm, however a great improvement can be made with the information of the log-posterior derivatives used along, as the hybrid algorithm appears to offer consistent but not spectacular gains over the Langevin algorithm, either with univariate or multivariate updating. Note also that there is no extra machine time required to complete the algorithm when the gains are achieved by random walk suppression with the GW or hybrid algorithm.

- **The comparison between univariate updating and multivariate updating**

All four algorithms with univariate updating are more efficient and converge more quickly than those with multivariate updating, separately. This is especially true to the RW and GW algorithms, which do not use the information of the log-posterior derivatives. Both algorithms offer a
mean reduction in relative error on the order of 50%, but about 25% for the Langevin and hybrid algorithms. For the speed of convergence, all algorithms with univariate updating offer a lot of gains over those with multivariate updating. As a trade-off, more computing time is needed to complete the algorithm.

In summary, the hybrid algorithm with univariate updating has the best performance and the guided walk algorithm with multivariate updating has the worst performance over all eight treatments (since it's a $2 \times 2 \times 2$ factorial Monte Carlo experiment), either in the sense of the mean $FIT$ statistic or the speed of convergence. In general, it seems that the efficiency and convergence time gains are achieved abundantly by univariate updating or/and by using log-posterior derivatives. As a trade-off, the extra human effort to understand and program the algorithm is needed, when using log-posterior derivatives, for example; and extra machine time required to complete the algorithm is needed when using univariate updating or using log-posterior derivatives. While those gains achieved by random walk suppression with the GW algorithm (only with univariate updating) or the hybrid algorithm might be viewed as moderate in an era of inexpensive computation, it should be noted that they come essentially for free.

The best acceptance rate varies from algorithm to algorithm. For example, the best acceptance rate for the RW, GW, Langevin and hybrid algorithm with univariate updating is 50%, 60%, 70%, and 80%, respectively. This suggests that the more efficient the algorithm is, the higher the acceptance rate is.
Figure 4.1: Comparisons of algorithms with one-at-a-time updating, based on a 5-variate normal distribution. The upper panel compares the mean FIT statistics for the four algorithm. The other four panels contain boxplots for the m = 200 FIT statistics obtained using the RW, GW, Langevin and hybrid algorithm, respectively, as a function of acceptance rate.
Figure 4.2: Comparisons of algorithms with one-at-a-time updating, based on a 20-variate normal distribution. The upper panel compares the mean FIT statistics for the four algorithms. The other four panels contain boxplots for the m = 200 FIT statistics obtained using the RW, GW, Langevin and hybrid algorithm, respectively, as a function of acceptance rate.
Figure 4.3: Comparisons of algorithms with all-at-a-time updating, based on a 5-variate normal distribution. The upper panel compares the mean FIT statistics for the four algorithm. The other four panels contain boxplots for the $m = 200$ FIT statistics obtained using the RW, GW, Langevin and hybrid algorithm, respectively, as a function of acceptance rate.
Figure 4.4: Comparisons of algorithms with all-at-a-time updating, based on a 20-variate normal distribution. The upper panel compares the mean FIT statistics for the four algorithms. The other four panels contain boxplots for the \( m = 200 \) FIT statistics obtained using the RW, GW, Langevin and hybrid algorithm, respectively, as a function of acceptance rate.
Figure 4.5: Comparisons of the speed of the convergence between the algorithms. Each panel contains boxplots of four algorithms with 200 of the same starting points from the uniform distribution on the hyper-square \((0, 20) \times \ldots \times (0, 20)\).
Chapter 5

A example: a gamma regression model

5.1 Introduction

In this chapter, a gamma regression model is applied to a data set on the volume of trees. The model is fitted by simulating a Markov chain which has the posterior distribution on the parameter space as its stationary distribution. First, Markov chains are constructed by using the random walk Metropolis, guided walk, Langevin and hybrid algorithms, respectively. Then, an MCMC convergence diagnostic is implemented on each algorithm to compare the speed of convergence between these four algorithms.
5.2 The model

In order to be able to estimate the volume of a tree from its diameter and height, a model is constructed for the specific structure of the data set, which was reported on by Ryan, Jointer and Ryan (1985). In particular, the data contains three variables: Diameter, Height and Volume, for 31 black cherry trees in Allegheny National Forest, Pennsylvania. The histograms of these three variables are given in Figure 5.1.

Thus, the response variable is Volume, denoted by $Y$, while the two predictor variables are Diameter and Height, denoted by $X_1$ and $X_2$, respectively. Our objective is to examine the relationship of $Y$ with $X_1$ and $X_2$ by using a statistical model. We assume that the observations from those trees should be uncorrelated. Note that the response variable $Y$ (Volume) is always positive and its histogram in Figure 1 indicates that the distribution of $Y$ is skewed to the right. This suggests that $Y$ might have a gamma distribution with a pdf:

$$f(y|r, \mu) = \mu^r y^{r-1} e^{-\mu y}/\Gamma(r),$$

where $r$ is the shape parameter, $\mu$ is the inverse scale parameter.

A generalized linear model could be constructed to fit the data. Note that we could consider the trees as cylinders, so that the volume would be

$$Y = \pi \left( \frac{X_1}{2} \right)^2 X_2,$$

by taking log’s for both sides we get

$$\log Y = \log(\pi/4) + 2 \log X_1 + \log X_2.$$
Thus, the log link function might be chosen, given by

$$\eta = \log E(Y) = \log(\tau/\mu).$$

The systematic component of the generalized linear model relates $\eta$ to the covariates Diameter and Height through a linear model:

$$\eta_i = \alpha + \beta_1 \log x_{i1} + \beta_2 \log x_{i2},$$

where $x_{i1}$ and $x_{i2}$ are the $i^{th}$ observed values of Diameter and Height, respectively. Hence, the likelihood function for the data is

$$f(y|\eta, \mu_i) = \prod_{i=1}^{31} \mu_i^{r_i} e^{-\mu_i y_i} / \Gamma(r),$$

where $y_i$ is the $i^{th}$ observed value of Volume, and $\mu_i = \tau e^{-\eta_i}$.

In order to do Bayesian estimation, a prior distribution is required. As is typically done for regression parameters, improper, locally uniform priors are specified for $\alpha$, $\beta_1$ and $\beta_2$. That is, constant prior densities with respect to Lebesgue measure are used. Note that the exponential distribution is a special case if the shape parameter $r$ is equal to 1 in the gamma distribution, then a standard normal distribution is specified as a prior for $\log(r)$, that is, the prior for the shape parameter $r$ is the log-normal distribution with the pdf

$$f(r) = \frac{1}{\sqrt{2\pi r}} e^{-(\log r)^2 / 2}.$$

The prior for the shape parameter is then 'centered' at the exponential model. Finally, the log-posterior for the data can be expressed as follows

$$l(y) = r \sum_{i=1}^{31} \log \mu_i + (r - 1) \sum_{i=1}^{31} \log y_i - \sum_{i=1}^{31} \mu_i y_i - 31 \log \Gamma(r) - 0.5(r \log r)^2 - \log r.$$
5.3 Re-parameterization

A very common problem in MCMC algorithms is high posterior correlations. High posterior correlations could cause the Markov chain to get stuck at a state for several iterations and thus the MCMC algorithm will mix slowly. The re-parameterization is a strategy designed to reduce posterior correlations in commonly used models. The remedy is to work with centered covariates in equation (1). That is, we substitute \( \log(x_{i1}) \) and \( \log(x_{i2}) \) in (1) by

\[
\log x_{i1} - \frac{1}{31} \sum_{i=1}^{31} \log x_{i1},
\]

and

\[
\log x_{i2} - \frac{1}{31} \sum_{i=1}^{31} \log x_{i2},
\]

respectively. After the re-parameterization the performances of all four algorithms are greatly improved. We initially use the link function

\[
\eta = \log(\mu),
\]

which is equivalent to the situation in which the parameter \( \alpha \) in (1) is re-parameterized by \( (\alpha + \log r) \). But the performances of the algorithms are not as good as those using the link function \( \eta = \log(r/\mu) \). In fact, the Markov chain is not convergent and the posterior correlation between \( \alpha \) and \( r \) is very high.
5.4 The first partial derivatives of the log posterior density

The Langevin algorithm and the hybrid algorithm require evaluations of the log unnormalized posterior density, along with its first partial derivatives. For simplicity, we assume that the data is already centered as in Section 5.3, and denote

\[ z_i = y_i - \alpha - \beta_1 \log x_{i1} - \beta_2 \log x_{i2}. \]

Then the first partial derivatives for \( \alpha, \beta_1, \beta_2 \) and \( r \) can be expressed as follows

\[
\frac{\partial l(y)}{\partial \alpha} = \sum_{i=1}^{31} r(e^{z_i} - 1), \\
\frac{\partial l(y)}{\partial \beta_1} = \sum_{i=1}^{31} r(e^{z_i} - 1) \log x_{i1}, \\
\frac{\partial l(y)}{\partial \beta_2} = \sum_{i=1}^{31} r(e^{z_i} - 1) \log x_{i2}, \\
\frac{\partial l(y)}{\partial r} = -1/r - (\log r)/r + \sum_{i=1}^{31} (\log r + 1 - \frac{d}{dr}\Gamma(r) + z_i - e^{z_i}).
\]

Note that an automatic routine, `digamma`, for the calculation of \( \frac{d}{dr}\Gamma(r) \) is available in SPLUS.

5.5 The MCMC convergence diagnostics of Gelman and Rubin

In practice, MCMC convergence diagnostics are much more difficult than in the simulation study in Chapter 4. First, the stationary distribution will
always be unknown to us, so this basic difficulty will plague the convergence
diagnostic. Secondly, the random walk could remain for many iterations in
a region heavily influenced by the starting distribution. So it might be not
possible to monitor convergence of an iterative simulation from a single chain
(i.e., one random walk).

The convergence diagnostics suggested by Gelman and Rubin (1992)
try to monitor convergence based on detecting when the Markov chains have
‘forgotten’ their starting points, by comparing several sequences drawn from
different starting points and checking that they are indistinguishable. That
is, multiple starting points are needed with finite-length sequences to avoid
inferences being unduly influenced by slow-moving realizations of the itera-
tive simulation. In particular, we label the $m$ parallel chains of length $n$ as
$(x_{ij}), j = 1, ..., n; i = 1, ..., m$, and we compute two quantities, the between-
chain variance $B$:

$$B = \frac{n}{m-1} \sum_{i=1}^{m} (\bar{x}_i - \bar{x})^2,$$

where $\bar{x}_i = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$, and $\bar{x} = \frac{1}{m} \sum_{i=1}^{m} \bar{x}_i,$

and the within-chain variances $W$:

$$W = \frac{1}{m} \sum_{i=1}^{m} s_i^2,$$

where $s_i^2 = \frac{1}{n-1} \sum_{j=1}^{n} (x_{ij} - \bar{x}_i)^2$.

Then the convergence is diagnosed when the variance between the different
sequences is no larger than the variance within each individual sequence. The
method involves two steps:

• Step 1 is to obtain an over-dispersed estimate of the target distribution
  and to generate from it the starting points for the desired number of
independent chains.

• Step 2 is to be carried out for each scalar quantity of interest after running the Markov chains for the desired number of iterations, say $2n$. It involves using the last $n$ iterations to reestimate the target distribution of scalar quantity as a conservative Student $t$ distribution, the scale parameter of which involves both the between-chain variance and within-chain variance. Convergence is monitored by estimating the factor by which the scale parameter might shrink if sampling were continued indefinitely, namely

$$
\sqrt{\hat{R}} = \sqrt{\left(\frac{n-1}{n} + \frac{m+1}{mn} B\right) \frac{df}{df - 2}}
$$

where $B$ is the variance between the means from the $m$ parallel chains, $W$ is the average of the $m$ within-chain variances, and $df$ is the degrees of freedom of the approximating $t$ density.

Gelman and Rubin recommended an iterative process of running additional iterations of the parallel chains and redoing step 2 until the 'shrink factor' for all quantities of interest are near 1. Gelman and Rubin suggest that the value of $\hat{R}$ greater than 1.2 indicates poor convergence. They interpreted the fact that the 'shrink factor' approaches 1 when the pooled within-chain variance dominates the between-chain variance to mean that at that point, all chains have escaped the influence of their starting points and have traversed all of the target distribution.
Table 5.1: Iterations to convergence with one-at-a-time updating:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accep. rate</th>
<th>$\alpha$</th>
<th>$\beta_1$</th>
<th>$\beta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random walk</td>
<td>0.50</td>
<td>160</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Guided walk</td>
<td>0.65</td>
<td>130</td>
<td>130</td>
<td>140</td>
</tr>
<tr>
<td>Langevin</td>
<td>0.85</td>
<td>120</td>
<td>130</td>
<td>140</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.90</td>
<td>80</td>
<td>90</td>
<td>90</td>
</tr>
</tbody>
</table>

5.6 Testing the speed of convergence

Once the Markov chains are constructed using the MCMC algorithms, we could use the convergence diagnostics of Gelman and Rubin to examine the convergence of the Markov chains. First, we need to assess the optimal acceptance rate for each algorithm. Thus, a series of proposal variances and a collection of seven overdispersed starting points are chosen. For each starting point, and each of a series of proposal variances, a MCMC chain of length $n = 4000$ is simulated. Then the 'shrink factor' is calculated for each proposal variance, using seven parallel Markov chains (i.e., corresponding to seven starting points). This could be carried out as follows. First, the chain for each variable of interest is split into a number of segments as follows: the first contains samples 1:50; the second contains samples 1:(50 + 10); the third contains samples 1:(50 + 20) and so on. Then the shrink factor is computed for each segment. The best proposal variance for each of the algorithms is the one for which the 'shrink factor' approaches one most quickly. The corresponding acceptance rate is then chosen as the best acceptance rate for the algorithm.

In the situation of 'one-at-a-time' updating, the 'shrink factor' ap-
Table 5.2: Iterations to Convergence with all-at-a-time updating:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accpt. rate</th>
<th>$\alpha$</th>
<th>$\beta_1$</th>
<th>$\beta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random walk</td>
<td>0.40</td>
<td>1000</td>
<td>1000</td>
<td>1200</td>
</tr>
<tr>
<td>Guided walk</td>
<td>0.50</td>
<td>1200+</td>
<td>1200+</td>
<td>1200+</td>
</tr>
<tr>
<td>Langevin</td>
<td>0.60</td>
<td>180</td>
<td>160</td>
<td>180</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.70</td>
<td>110</td>
<td>110</td>
<td>105</td>
</tr>
</tbody>
</table>

approaches one most quickly when the acceptance rate is 50% for the RW algorithm, 65% for the GW algorithm, 85% for the Langevin algorithm, and 90% for the hybrid algorithm. For a collection of seven overdispersed starting points, multiple MCMC chains of length $n = 1000$ are simulated for each algorithm, using these respective acceptance rates. Each chain is labeled as having converged once the value of the shrink factor for each variable is smaller than 1.2. The numbers of iterations to convergence (i.e., $\hat{R}$ is smaller than 1.2) for $\alpha$, $\beta_1$ and $\beta_2$ are around 45 for the hybrid algorithm, around 65 for both the Langevin and GW algorithms, and around 100 for the RW algorithm. The plot of the shrink factor versus last iteration in segment for each regression parameter is given in Figure 5.2, 5.3 and 5.4, respectively. This implies that the algorithms using log-posterior derivatives tends to converge more quickly than those not using log-posterior derivatives. Also, the hybrid algorithm converges a bit more quickly that the Langevin algorithm and the GW algorithm converges a bit more quickly that the RW algorithm. Thus the random walk behavior is suppressed in both the GW and hybrid algorithms. The summary of results about the speed of convergence is given in Table 5.1.

In the situation of ‘all-at-a-time’ updating, the ‘shrink factor’ approaches
one most quickly when the acceptance rate is 40% for the RW algorithm, 50% for the GW algorithm, 60% for the Langevin algorithm, and 70% for the hybrid algorithm. For a collection of seven overdispersed starting points, multiple MCMC chains of length $n = 1000$ are simulated for each algorithm, using these respective acceptance rates. Each chain is labeled as having converged once the value of the shrink factor for each variable is smaller than 1.2. The plot of the shrink factor versus last iteration in segment for each regression parameter is given in Figure 5.5, 5.6 and 5.7, respectively. The summary of results about the speed of convergence is given in Table 5.2. This results imply that the algorithms using log-posterior derivatives tends to converge much more quickly than those not using log-posterior derivatives. But the performance of the GW algorithm is not as good as in the situation of 'one-at-a-time'. In fact, the GW algorithm does not suppress the random walk behavior, which is identical to the performance of the GW algorithm with 'all-at-a-time' updating in the simulation study in Chapter 3. Note that similarly as in the situation of 'one-at-a-time', the hybrid algorithm converges more quickly than the Langevin algorithm, which implies that, with the information of the log-posterior derivatives used, the hybrid algorithm does suppress the random walk behavior.

For the parameter updating comparison between 'one-at-a-time' and 'all-at-a-time', all four algorithms with 'one-at-a-time' updating converge more quickly than those with 'all-at-a-time' updating, with more computing time needed to complete the algorithm. In particular, the RW and GW algorithms
Table 5.3: Estimation of the regression parameter with the Monte Carlo standard error (one-at-a-time updating):

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$\alpha$ (s.e.)</th>
<th>$\beta_1$ (s.e.)</th>
<th>$\beta_2$ (s.e.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random walk</td>
<td>3.28(0.0014)</td>
<td>1.99(0.01)</td>
<td>1.10(0.026)</td>
</tr>
<tr>
<td>Guided walk</td>
<td>3.28(0.001)</td>
<td>1.99(0.01)</td>
<td>1.12(0.026)</td>
</tr>
<tr>
<td>Langevin</td>
<td>3.28(0.0002)</td>
<td>2.0(0.001)</td>
<td>1.07(0.003)</td>
</tr>
<tr>
<td>Hybrid</td>
<td>3.28(0.0005)</td>
<td>1.99(0.002)</td>
<td>1.11(0.004)</td>
</tr>
</tbody>
</table>

Table 5.4: Estimation of the regression parameter with the Monte Carlo standard error (all-at-a-time updating):

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$\alpha$ (s.e.)</th>
<th>$\beta_1$ (s.e.)</th>
<th>$\beta_2$ (s.e.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random walk</td>
<td>3.35(0.19)</td>
<td>2.16(0.62)</td>
<td>1.07(0.50)</td>
</tr>
<tr>
<td>Guided walk</td>
<td>3.57(1.30)</td>
<td>1.70(1.28)</td>
<td>1.43(1.32)</td>
</tr>
<tr>
<td>Langevin</td>
<td>3.28(0.003)</td>
<td>1.97(0.002)</td>
<td>1.15(0.004)</td>
</tr>
<tr>
<td>Hybrid</td>
<td>3.28(0.002)</td>
<td>1.98(0.006)</td>
<td>1.12(0.002)</td>
</tr>
</tbody>
</table>

appear to offer much more gains than the Langevin and hybrid algorithms.

Monte Carlo standard errors of the regression parameters can be calculated over the multiple MCMC chains as follows. For each starting point, the estimation of the regression parameter is obtained from the Markov chain, using the best acceptance rate for the algorithm. Then the standard error of this regression parameter estimation can be calculated over a collection of different starting points, and we refer to this standard error as the Monte Carlo standard error. In general, one expects algorithms which get smaller Monte Carlo standard error to be more efficient. In the example, the estimations of the regression parameters $\alpha$, $\beta_1$ and $\beta_2$ are obtained over a collection of seven overdispersed starting points for each algorithm, after a burn-in period for the
algorithm to converge. Thus the Monte Carlo standard errors for the estimations of the regression parameters can be calculated over this collection of seven starting points. We summarize the results with 'one-at-a-time updating' in Table 5.3, and those with 'all-at-a-time' updating in Table 5.4. In the case of 'one-at-a-time', the algorithms which use log-posterior derivatives achieve smaller Monte Carlo standard error and therefore are more efficient than the algorithms which don't; but there are no big differences either between the RW and GW algorithms, or between the Langevin and hybrid algorithms. In the case of 'all-at-a-time', for the comparison of 'with derivative' updating against 'without derivative' updating, we have the same result as in the case of 'one-at-a-time', i.e., the algorithms with derivative updating are more efficient than the algorithms without derivative updating; but there are no big differences between the Langevin and hybrid algorithms. Finally, the GW algorithm with 'all-at-a-time' updating fails to suppress the random walk behavior and is less efficient than the RW algorithm.
Figure 5.1: Histograms of the three variables of the tree data
Figure 5.2: Comparison of the speed of convergence for one-at-a-time updating, based on how fast the shrink factor for $\alpha$ approaches one.
Figure 5.3: Comparison of the speed of convergence for one-at-a-time updating, based on how fast the shrink factor for $\beta_1$ approaches one.
Figure 5.4: Comparison of the speed of convergence for one-at-a-time updating, based on how fast the shrink factor for $\beta_2$ approaches one.
Figure 5.5: Comparison of the speed of convergence for all-at-a-time updating, based on how fast the shrink factor for $\alpha$ approaches one.
Figure 5.6: Comparison of the speed of convergence for all-at-a-time updating, based on how fast the shrink factor $\beta_1$ approaches one.
Figure 5.7: Comparison of the speed of convergence for all-at-a-time updating, based on how fast the shrink factor $\beta_2$ approaches one.
Chapter 6

Discussion

The random walk Metropolis, guided walk, Langevin and hybrid algorithms have been compared based on (i) simulation studies using multivariate normal target distributions in Chapter 4 and (ii) a posterior distribution for parameters given observed data in Chapter 5. Our summaries in Chapter 4 and Chapter 5 show that both results in Chapter 4 and in Chapter 5 are very similar. First, all four algorithms converge more quickly by using 'one-at-a-time' updating than using 'all-at-a-time' updating. Second, the algorithms using derivatives converge more quickly than those not using derivatives. The rationale is the algorithms using derivatives use more information about the posterior. Third, the guided walk algorithm suppresses the random walk behavior only in the situation of the 'one-at-a-time' updating. Our interpretation is that there are many combinations on the signs of the components of the latent vector $P$, hence it is not as meaningful as in the 'one-at-a-time' to 'guide' the direction once a candidate state is rejected. However, with the informa-
tion of log-posterior derivatives used, the hybrid algorithm does suppress the random walk behavior in both two different updating parameter schemes. The rationale here is not very clear to us and further research could be helpful. But it can be shown that, with $N$ energy gradient evaluations, the hybrid algorithm can move a distance proportional to $N$, while a random algorithm will likely have traversed only a distance proportional to $\sqrt{N}$ (Neal, 1993). In summary, an algorithm which uses more information about posterior or/and avoids more random walk behavior is likely to be more efficient. For example, of these four MCMC algorithms, the hybrid algorithm performs best because (i) it uses the information of log posterior derivatives and (ii) it can also avoid some of the random walk behavior characteristic of simpler versions of the Metropolis algorithm. In addition, the hybrid algorithm is also not too difficult to implement, provided that the kernel of the log posterior density and its first partial derivatives can be evaluated.

We also saw that the best acceptance rate varies from algorithm to algorithm. From our results in Chapter 4 and 5, the more efficient the algorithm is, the higher the acceptance rate is. As a result, the chain moves back and forth across the distribution and converges more quickly.

It is important to keep in mind that the criteria for detecting the convergence in Chapter 4 are different from those in Chapter 5. In the simulation study, the target posterior distributions are multivariate normal distributions. Hence a $\chi^2$ goodness of fit statistic can be constructed to measure how well a simulated chain represents the true distribution, given that the chain is
started in the true distribution. This comprises a major issue in the comparisons between algorithms in the simulation study in Chapter 4. Another issue of comparisons between algorithms in chapter 4 is the speed of convergence. Given the real target distribution, we can sample the initial values of the parameters from another distribution to make it challenging for the algorithm to converge and this comprises our second issue in the comparisons between algorithms in the simulation study in Chapter 4. However, these priorities are no longer available in the application of a real data set in Chapter 5 because the ‘real’ target distribution is unknown. Instead of using a $\chi^2$ goodness of fit statistic, the approach here to monitoring convergence is based on several sequences drawn from different starting points, then we identify ‘convergence’ in a sense that the empirical distribution of simulations obtained separately from each sequence is approximately the same as the distribution obtained by mixing all the sequences together. Note that in addition to the convergence diagnostics of Gelman and Rubin (1992), there are many other MCMC convergence diagnostics available from the existing literature. Cowles and Carlin (1996) provide an expository review of thirteen convergence diagnostics, which include Gelman and Rubin (1992), Raftery and Lewis (1992), Geweke (1992), Roberts (1992) and Yu and Mykland (1994). In general, different diagnostics attempt to examine some aspects of MCMC convergence.

With the efficiency and convergence time gains achieved by ‘one-at-a-time’ updating and/or using derivatives, we also witnessed that much more computing time is needed to complete the algorithm with ‘one-at-a-time’ up-
dating rather than the algorithm with 'all-at-a-time' updating; the extra hu-
man effect with more computing time to complete the algorithm is also needed
to understand and program the algorithm using derivatives rather than that
not using derivatives. But those gains achieved by random walk suppression
with the GW algorithm (one-at-a-time updating) or hybrid algorithm come
essentially for free.

In addition, when an algorithm using derivatives was used to construct
a Markov chain in the application of the tree data set in Chapter 5, we also saw
that the algorithm might have a problem when the value of the derivative was
too large at a specific point so that the computer just assign this value to be
'infinity', which could stop the algorithm before its completion. An immediate
remedy we have tried for this problem was to choose another starting point
and then run the algorithm again to try to avoid this specific point. Our
results reported in Chapter 5 are based on this method. Alternatively, another
approach to this problem could be to substitute $g(x)$ in the algorithm in Section
3.5 by

$$g(x) = \begin{cases} 
\nabla \log \pi(x) & -C < \nabla \log \pi(x) < C, \\
C & \nabla \log \pi(x) > C, \\
-C & \nabla \log \pi(x) < -C.
\end{cases} \tag{1}$$

where $C$ is some constant. Hence the hybrid algorithm for updating from
$(X^t, P^t) = (x, p)$ to $(X^{t+1}, P^{t+1}) = (x', p')$ proceeds as

$$x^* \leftarrow x + \epsilon \{p + (\epsilon/2)g(x)\}$$

$$p^* \leftarrow -p - (\epsilon/2)\{g(x) + g(x^*)\}$$
\[ \alpha \leq \min \left\{ \frac{\pi(x^*, p^*)}{\pi(x, p)}, 1 \right\} \]

\[ (x', p') \left\{ \begin{array}{l} (x^*, p^*) \text{ with probability } \alpha, \\ (x, p) \text{ with probability } 1 - \alpha. \end{array} \right. \]

\[ p' \leftarrow -p' \]

\[ p' \leftarrow \rho p' + (1 - \rho^2)^{1/2} z \text{ where } z \sim N(0, I), \]

where \( g(x) \) is determined from (1). It should be noted that the property of time reversibility is still hold after the modification. Thus, similarly as the previous version of hybrid algorithm, no adjustment factor is required in the Metropolis-Hastings acceptance probability. That is, with probability \( \alpha \) the new state is set equal to the candidate state \((x^*, p^*)\), and with probability \( 1 - \alpha \) the new state is set equal to the old state \((x, p)\). This method was also tested for the same data set as in Chapter 5, and it worked well but the detailed results are not reported since the algorithm of interest in this thesis is the original version as discussed in Chapter 3.

For the model of the tree data set in Chapter 5, a 'mixing' algorithm with 'one-at-a-time' updating of the parameters was also tested. In particular, the shape parameter \( r \) in the gamma density was updated by using the guided walk algorithm, but the regression parameters \( \alpha, \beta_1 \) and \( \beta_2 \) were updated by using hybrid algorithm. For this example, the performance of this 'mixing' algorithm was virtually identical to that of the hybrid algorithm with 'one-at-a-time' updating the parameters. Consequently detailed results are not reported. There is no existing literature to support this 'mixing' algorithm and this algorithm was not pursued further.
Markov chain Monte Carlo has been an important tool for the analysis of complex statistical problems. Our main purpose of this study is to try to make a contribution toward a better understanding of the differences between those four algorithms' performances. However, different Markov chains have different characteristics in different problems. In our study, the algorithm comparisons are based on simulation study using only multivariate normal distributions, which allows us easily to construct a $\chi^2$ goodness of fit statistic which measures how well a simulated chain represents the target distribution. To better understand and assess the performances of different MCMC algorithms, more scenarios for comparing the algorithms involved other more target distributions are clearly needed. Also, those algorithms should be tested using different posterior distributions for parameters given observed data in applications.
Bibliography


