A COMPARISON OF ALGORITHMS FOR SAMPLING FROM A DISTRIBUTION

by

Na Lei
B.Sc., University of Science and Technology, China, 2001
M.Sc., University of Science and Technology, China, 2004

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the Department
of
Statistics and Actuarial Science

© Na Lei 2007
SIMON FRASER UNIVERSITY
2007

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without the permission of the author.

APPROVAL

Name:	Na Lei
Degree:	Master of Science
Title of project:	A Comparison of Algorithms for Sampling from a Distribution
Examining Committee:	Dr. Joan Hu Chair
	Dr. Randy Sitter Senior Supervisor Simon Fraser University
	Dr. Derek Bingham Simon Fraser University
	Dr. Tom Loughin External Examiner Simon Fraser University
Date Approved:	Aug 20, 2007

SFU SIMON FRASER UNIVERSITY
LIBRARY

Declaration of Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the "Institutional Repository" link of the SFU Library website <www.lib.sfu.ca> at: <http://ir.lib.sfu.ca/handle/1892/112>) and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

While licensing SFU to permit the above uses, the author retains copyright in the thesis, project or extended essays, including the right to change the work for subsequent purposes, including editing and publishing the work in whole or in part, and licensing other parties, as the author may desire.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, BC, Canada

Revised: Fall 2007

Abstract

Sampling from a distribution is an active problem in statistics. When the distribution is easy to sample from, methods like Monte Carlo are applicable. But when the distribution is complex, of non-standard form or multivariate, more complicated algorithms are required. The well-known Markov Chain Monte Carlo method using the Metropolis-Hastings (MH) algorithm can perform very well to sample the complicated distributions in many situations. But it has the drawback of being sensitive to the scale of the proposal distribution used. Recently, some algorithms have been introduced in the literature to avoid some of the problems of the MH algorithm. These include Graves method, Sliced sampling, and Equi-energy sampling. In this project, a simulation study is done to compare the performance of these algorithms under various settings of their tuning parameters when applied to various types of distributions.

Keywords: Bayesian Methods; Robust Parameter Design; Sampling; Stationary distribution

Subject Terms: Sampling; Monte Carlo Method; Regression analysis; Experimental design

Acknowledgments

Three years of studying in the statistics department of Simon Fraser University is a precious experience in my life.

I would first like to give my gratitude to my supervisor, Randy Sitter. Thank you for leading me in the statistics field and encouraging me to keep going when I met difficulty. Without your inspiring discussions and patient guidance, I would not have got over many hard points of this project.

I would like to thank Dr. Derek Bingham and Dr. Tom Loughin for the careful reviewing and insightful comments. I also want to give my gratitude to the instructors who taught me various courses and showed me various statistical problems: Dr. Boxin Tang, Dr. Joan Hu, Dr. Tim Swartz, Dr. Charmaine Dean, and Mr. Ian Bercovitz. Thank you also to the fellow graduate classmates for being company and growing together with me during my stay at SFU, and Kelly, Charlene and Sadika for your help always. Special gratitude is given to Crystal for taking the time to help me with this project.

Many thanks to Mark for the support and helping me to see the importance of confidence and optimism. My forever and unchangeable gratitude is given to my Mom and Dad. It is your love that supports me and keeps me going in the past, now, and in the future.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Sampling from a distribution is an active statistical problem. The Metropolis-Hastings (MH) algorithm has been given a lot of attention as a method of solving this sampling problem. It has the advantage of being able to deal with distributions that are complex, of non-standard form, and/or multivariate. But there are some problems in the application of the MH algorithm. For example, the algorithm can yield sample histograms that are very different from the true distribution if the tuning parameter, the step-width, is not well-chosen, especially when the true distribution is multi-modal. In recent years, some new methods have been introduced to try to avoid some of the problems that the MH algorithm encounters. These include "slice sampling" (Neal, 2003), and the Equi-energy (EE) sampling algorithm (Kou, Zhou, and Wong, 2006). In this project, we will consider these new methods, along with the MH algorithm, and compare their performances via simulation.

One main application of these methods is in drawing from the posterior distribution in a Bayesian framework. Thus, for motivational purposes, we will briefly outline Bayesian analysis so as to place the main contributions in this project within a broader framework.

Let $\boldsymbol{\theta} = (\theta_1, ..., \theta_p)'$ denote a vector of unknown parameters in the distribution of $\boldsymbol{y} = (y_1, ..., y_n)'$, denoted $f(\boldsymbol{y}|\boldsymbol{\theta})$. In a Bayesian framework, parameters $\boldsymbol{\theta}$ are considered to be random variables with distribution $g(\boldsymbol{\theta})$. This is termed the prior distribution and represents *a priori* beliefs concerning the parameters, before data collection. As such, $f(\boldsymbol{y}|\boldsymbol{\theta})$ becomes the likelihood given $\boldsymbol{\theta}$. Then via Bayes' Rule, the *posterior* distribution of $\boldsymbol{\theta}$ is

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto f(\boldsymbol{y}|\boldsymbol{\theta})g(\boldsymbol{\theta}).$$

The posterior distribution is interpreted to represent updated beliefs about $\boldsymbol{\theta}$ after data collection, and is used for inference about $\boldsymbol{\theta}$.

Estimation of $\boldsymbol{\theta}$ is often done via examining various quantities based on the posterior distribution. For example, a common point estimate of a function $h(\boldsymbol{\theta})$ is its posterior mean,

$$E[h(\boldsymbol{\theta})|\boldsymbol{y}] = \int h(\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\boldsymbol{y})d\boldsymbol{\theta}.$$

In practice, these integrals are often intractable and are evaluated numerically using Monte Carlo techniques. That is, let $\boldsymbol{\theta}^{(1)}, \cdots, \boldsymbol{\theta}^{(N)}$ be a large number of draws from $\pi(\boldsymbol{\theta}|\boldsymbol{y})$; then

$$E[h(\boldsymbol{\theta})|\boldsymbol{y}] \doteq \frac{1}{N}\sum_{i=1}^{N} h(\boldsymbol{\theta}^{(i)}).$$

Such posterior draws can also be used to examine other properties of the posterior distribution of $\boldsymbol{\theta}$. For example, one can construct credible intervals or various percentiles.

In simple cases, such as when one can directly sample from the distribution, Monte Carlo methods can work well. However, if the posterior distribution is complex or of non-standard form, more sophisticated methods are required. For example, sometimes $\int f(\boldsymbol{y}|\boldsymbol{\theta})g(\boldsymbol{\theta})d\boldsymbol{\theta}$ is hard to calculate, which makes $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ have a complicated form, because

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{f(\boldsymbol{y}|\boldsymbol{\theta})g(\boldsymbol{\theta})}{\int f(\boldsymbol{y}|\boldsymbol{\theta})g(\boldsymbol{\theta})d\boldsymbol{\theta}}.$$

This can essentially imply that we only know $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ is proportional to $f(\boldsymbol{y}|\boldsymbol{\theta})g(\boldsymbol{\theta})$, a non-standard form. One such method for sampling from complicated forms is the Metropolis-Hastings (MH) Algorithm. It was developed by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller (1953), and subsequently generalized by Hastings (1970).

The MH algorithm was the first of the well known Markov Chain Monte Carlo (MCMC) methods. These methods have been actively applied in combination with Bayesian analysis. MCMC methods work as follows. Suppose we can generate a sequence of $\boldsymbol{\theta}$, $\{\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots\}$ such that, at each iteration $i \geq 0$, the next state $\boldsymbol{\theta}^{(i+1)}$ is sampled from a distribution $P(\boldsymbol{\theta}^{(i+1)}|\boldsymbol{\theta}^{(i)})$ which depends only on the current state of the chain, $\boldsymbol{\theta}(i)$, i.e. it is a Markov chain. As i increases, the chain will gradually "forget" its initial state $\boldsymbol{\theta}^{(0)}$ and converge to a stationary distribution $P(\boldsymbol{\theta}|\boldsymbol{y})$, which does not depend on i or $\boldsymbol{\theta}^{(0)}$. A properly constructed chain will have stationary distribution $\pi(\boldsymbol{\theta}|\boldsymbol{y})$. After a sufficiently large number of *burn-in* (say B) iterations, points $\{\boldsymbol{\theta}^{(i)} : i = B+1, \cdots, n\}$ will be samples (approximately) from

$\pi(\boldsymbol{\theta}|\boldsymbol{y})$. Thus, if $\{\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots \boldsymbol{\theta}^{(N)}\}$ are draws generated by such a chain, the posterior mean of the function $h(\boldsymbol{\theta})$ can be estimated as

$$E[h(\boldsymbol{\theta})|\boldsymbol{y}] \doteq \frac{1}{N}\sum_{i=1}^{N} h(\boldsymbol{\theta}^{(i)}).$$

Given a previous point, $\boldsymbol{\theta}^{(i)}$, the MH algorithm generates a new candidate point $\boldsymbol{\theta}^*$ from a proposal distribution $q(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{(i)})$ which is a conditional distribution given $\boldsymbol{\theta}^{(i)}$. For example, assume the unknown parameter is σ and the proposal distribution is a uniform distribution. Then the candidate point σ^* is generated from a $Unif(\sigma^{(i)} - W, \sigma^{(i)} + W)$, where W is a constant typically called the "stepwidth". That is, $q(\sigma^*, \sigma^{(i)}) = (2W)^{-1} I_{[\sigma^{(i)} - W \le \sigma^* \le \sigma^{(i)} + W]}$, where $I_{[\cdot]}$ is the indicator function. The candidate point is accepted with some user-specified probability α,

$$\alpha = \alpha(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^*) = \min\left[\frac{\pi(\boldsymbol{\theta}^*|\boldsymbol{y})q(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{(i)})}{\pi(\boldsymbol{\theta}^{(i)}|\boldsymbol{y})q(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^*)}, 1\right].$$

The reason that the MH algorithm can deal with non-standard distributions lies in α. Since the ratio of $\pi(\boldsymbol{\theta}^*|\boldsymbol{y})$ and $\pi(\boldsymbol{\theta}^{(i)}|\boldsymbol{y})$ is not affected even when $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ is of non-standard form, say $f(\boldsymbol{y}|\boldsymbol{\theta})g(\boldsymbol{\theta})$,

$$\frac{\pi(\boldsymbol{\theta}^*|\boldsymbol{y})}{\pi(\boldsymbol{\theta}^{(\mathbf{i})}|\boldsymbol{y})} = \frac{f(\boldsymbol{y}|\boldsymbol{\theta}^*)g(\boldsymbol{\theta}^*)}{f(\boldsymbol{y})} \Big/ \frac{f(\boldsymbol{y}|\boldsymbol{\theta}^{(\mathbf{i})})g(\boldsymbol{\theta}^{(\mathbf{i})})}{f(\boldsymbol{y})} = \frac{f(\boldsymbol{y}|\boldsymbol{\theta}^*)g(\boldsymbol{\theta}^*)}{f(\boldsymbol{y}|\boldsymbol{\theta}^{(\mathbf{i})})g(\boldsymbol{\theta}^{(\mathbf{i})})}$$

and thus α does not depend on the possibly difficult to obtain $f(\boldsymbol{y}) = \int f(\boldsymbol{y}|\boldsymbol{\theta})g(\boldsymbol{\theta})d\boldsymbol{\theta}$.

Another well-known MCMC method is the Gibbs sampler. It was originally developed by Geman and Geman (1984), and was popularized by Casella and George (1992). Later this algorithm was shown to be a special case of the MH algorithm (Robert and Casella (1999)).

The Gibbs sampler requires the full conditional distribution of each element of $\boldsymbol{\theta}$ to be calculated. That is, the distribution of θ_j given all other elements of $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)'$, for each j. The simulation starts from an initial value $\boldsymbol{\theta}^{(0)}$. Given the i-th draw, $\boldsymbol{\theta}^{(i)}$, the next draw is generated by simulating each element of $\boldsymbol{\theta}$ in turn from their full conditional distribution,

$$\theta_1^{(i+1)} \sim \pi(\theta_1|\theta_2^{(i)}, \cdots, \theta_p^{(i)}),$$

$$\vdots$$

$$\theta_p^{(i+1)} \sim \pi(\theta_p|\theta_1^{(i)}, \cdots, \theta_{p-1}^{(i)}).$$

The simulation stops when sufficient samples have been collected.

Therefore, if we can calculate the full conditional distribution of each element of $\boldsymbol{\theta}$, and it has a form that can be directly sampled from, the Gibbs sampler is a viable strategy. However, if the full conditional distributions are difficult to calculate, or even if such is possible, they have a complex form rather than being a well-known distribution, other sampling strategies, such as the MH algorithm, are needed.

In Chapter 2 of this project, we use an example to demonstrate the MH algorithm and highlight problems it can encounter. Then some more recent methods that have been introduced in the literature to avoid some of these problems are introduced. A simulation study to compare the various methods is performed in Chapter 3. Further exploration on some interesting cases discovered in the simulation study are discussed in Chapter 4. Chapter 5 summarizes the work and discusses recommendations.

Chapter 2

The Algorithms

2.1 Metropolis Hastings Algorithm

In the previous Chapter, we briefly introduced the MH algorithm in a Bayesian framework. The MH algorithm has been frequently used to sample complex distributions in a broader context. Here we describe the algorithm in detail in a general form. Application to Bayesian analysis remains an important special case.

Assume the target distribution is $\pi(\boldsymbol{x})$. Given the previous point $\boldsymbol{x}^{(i)}$, a candidate point $\boldsymbol{x}^*$ is generated from a proposal distribution $q(\boldsymbol{x}^*, \boldsymbol{x}^{(i)})$. The MH algorithm is summarized as follows.

The Metropolis-Hastings Algorithm.

Step 1. Start from an initial value $\boldsymbol{x}^{(0)}$.

Step 2. Generate a proposal $\boldsymbol{x}^*$ from some proposal distribution $q(\boldsymbol{x}^*, \boldsymbol{x}^{(i)})$.

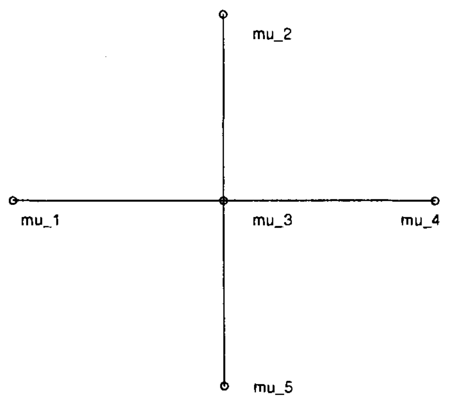
Step 3. Compute

$$\alpha(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*) = \min\left[\frac{\pi(\boldsymbol{x}^*)q(\boldsymbol{x}^*, \boldsymbol{x}^{(i)})}{\pi(\boldsymbol{x}^{(i)})q(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*)}, 1\right].$$

Step 4. With probability $\alpha(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*)$, let $\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^*$. Otherwise, let $\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)}$.

Step 5. Repeat Steps 2-4 until a sufficiently large sample is collected.

Figure 2.1: Positions of the mean vectors



The MH algorithm is an implementation of MCMC. The transition probability from the previous state to the next state is $P(\boldsymbol{x}^{(i+1)}|\boldsymbol{x}^{(i)}) = q(\boldsymbol{x}^{(i+1)}, \boldsymbol{x}^{(i)})\alpha(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i+1)})$. It satisfies the reversibility condition

$$\pi(\boldsymbol{x}^{(i+1)})P(\boldsymbol{x}^{(i+1)}|\boldsymbol{x}^{(i)}) = \pi(\boldsymbol{x}^{(i)})P(\boldsymbol{x}^{(i)}|\boldsymbol{x}^{(i+1)})$$

that assures the samples converge to a stationary distribution equal to $\pi(\boldsymbol{x})$. For discussion see Chib and Greenberg (1995).

Various densities can be chosen as the proposal distribution. One common family is a *random walk* where the candidate $\boldsymbol{x}^*$ equals $\boldsymbol{x}^{(i)}$ plus noise. The uniform distribution is such a commonly used choice. For example, each component $x_j^{(i+1)}$ of $\boldsymbol{x}^{(i+1)}$ is independently generated from a $Unif(x_j^{(i)} - W, x_j^{(i)} + W)$ centered at $x_j^{(i)}$, where W is a constant, and $x_j^{(i)}$ is the corresponding component of the previous state $\boldsymbol{x}^{(i)}$. Thus, $x^* = x^{(i)} + z$ where $z \sim Unif(-W, W)$. One characteristic of the uniform distribution being the proposal distribution is, since it is symmetric, $q(\boldsymbol{x}^*, \boldsymbol{x}^{(i)}) = q(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*)$ and cancel in $\alpha(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*)$, yielding

$$\alpha(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*) = \min\left[\frac{\pi(\boldsymbol{x}^*)}{\pi(\boldsymbol{x}^{(i)})}, 1\right].$$

In addition to choosing the family of proposal distributions, the choice of the spread of the specific proposal distribution used is very important. The choice of spread (or the step-widths) of the proposal distribution will greatly impact the performance of the MH algorithm. We use the following example to explain.

Example 1. Assume the target distribution we want to sample from is a mixture of spherical bivariate normal distributions,

$$\pi(\boldsymbol{x}) = \sum_{l=1}^{m} \frac{w_l}{2\pi\sigma_l^2} \exp\left\{ \frac{1}{2\sigma_l^2}(\boldsymbol{x}-\boldsymbol{\mu}_l)'(\boldsymbol{x}-\boldsymbol{\mu}_l) \right\},$$

where $\boldsymbol{x} = (x_1, x_2)'$, $m = 5$ is the number of bivariate normal distributions, $\sigma_l = 0.5$ is the common standard deviation for each element of the l-th bivariate normal distribution, $w_l = 1/5$ are the weights assigned to each distribution, and $\boldsymbol{\mu}_l = (\mu_{1l}, \mu_{2l})'$ are the mean vectors. Five mean vectors are positioned as a cross (see the Figure 2.1). Along the cross, the distance between two mean vectors is 3. The simulation starts from the initial point $\boldsymbol{x}^{(0)} = (-3, -3)'$.

In the implementation of the MH algorithm we use a uniform distribution centered at the current $\boldsymbol{x}$ as the proposal distribution, i.e.

$$q(\boldsymbol{x}^*, \boldsymbol{x}^{(i)}) = q_0(x_1^*, x_1^{(i)})q_0(x_2^*, x_2^{(i)}),$$

where

$$q_0(x_j^*, x_j^{(i)}) = \frac{1}{2W} I_{[x_j^{(i)}-W \le x_j^* \le x_j^{(i)}+W]} \text{ for } j = 1, 2,$$

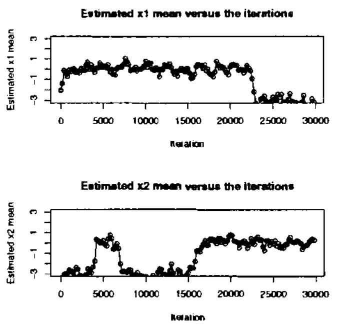
and W, the step-width, is set to three different values, 0.1, 3 and 10. For each step-width, the MH algorithm is run for 30000 iterations.

For each step-width, we let the simulation run 200, 400, $\cdots$, 30000, iterations and record the estimated mean of x_1 and x_2 (over each subsequent group of 200 iterations) versus the number of iterations. When the estimated mean becomes stable over iterations, that is the estimated mean change in a stable range, it corresponding iteration would typically be used as the approximate burn-in period. The histogram of the generated points are plotted to compare with the known marginal distributions. Figures 2.2-2.4 are the "variate mean vs. iteration" plots and "histogram" plots corresponding to step-widths $W = 0.1$, $W = 3$, and $W = 10$, respectively.

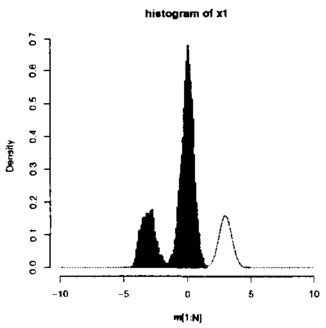
Viewing the histograms for step-width $W = 0.1$ (Figure 2.2), we see that the algorithm has a difficult time migrating from one hump to another. Viewing the "variate mean vs. iteration" plots, we see that the the mean of x_1 or x_2 stay in -3 for a period of time, then change dramaticly to 0, which shows the sampling move from one hump to another.

Viewing the histograms for step-width $W = 3$ (Figure 2.3), we see that the algorithm no longer has the difficulties observed for the smaller W, and seems to fit the marginal

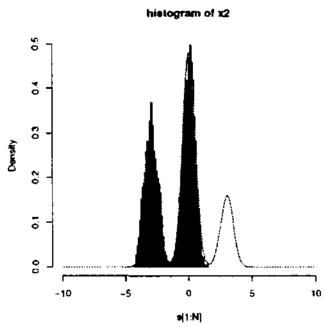
Figure 2.2: W=0.1



(a) variate mean vs. iteration

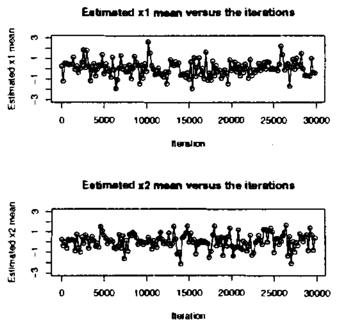


(b) x1 histogram

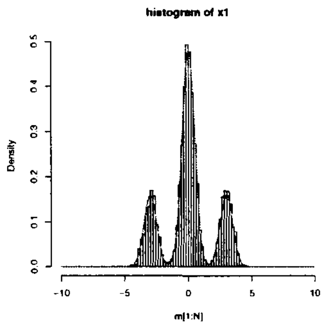


(c) x2 histogram

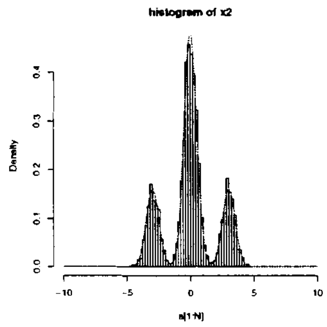
Figure 2.3: W=3



(a) variate mean vs. iteration

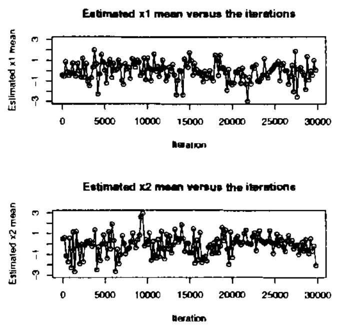


(b) x1 histogram

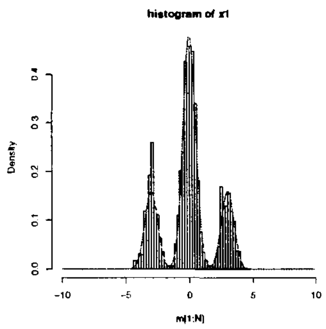


(c) x2 histogram

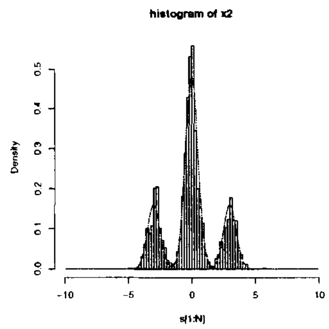
Figure 2.4: W=10



(a) variate mean vs. iteration



(b) x1 histogram



(c) x2 histogram

distributions very well. In addition, viewing the "variate mean vs. iteration" plots, we see that the estimated means of x_1 and x_2 quickly reached 0 and stayed in the stable vibration range.

Viewing the histograms for step-width $W = 10$ (Figure 2.4), we see that the algorithm seems to fit the marginal distributions fairly well, but comparing to $W = 3$, the smoothness of the fit is worse. Viewing the "variate mean vs. iteration" plots, we see that, as with $W = 3$, the estimated means of x_1 and x_2 also quickly reached 0 and stayed in some stable vibration range. Comparing with $W = 3$, the vibration range is bigger. For both $W = 3$ and $W = 10$, the burn-in can be set less or around 2000.

To see the reason that the algorithm's performance is impacted by choice of step-width, consider the acceptance probability of a candidate point $\boldsymbol{x}^*$ given the previous point $\boldsymbol{x}^{(i)}$,

$$\alpha(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*) = \min \left[\frac{\pi(\boldsymbol{x}^*)}{\pi(\boldsymbol{x}^{(i)})}, 1 \right].$$

Imagine we sample the points near a mode. If $\pi(\boldsymbol{x}^*) > \pi(\boldsymbol{x}^{(i)})$, $\alpha(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*) = 1$, which means the algorithm will always accept the candidate point. On the other hand, if $\pi(\boldsymbol{x}^*) < \pi(\boldsymbol{x}^{(i)})$, the algorithm will accept the candidate point with probability $\alpha(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*) = \frac{\pi(\boldsymbol{x}^*)}{\pi(\boldsymbol{x}^{(i)})}$. So when the step-width is large, it is more likely the candidate point, $\boldsymbol{x}^*$ is far from $\boldsymbol{x}^{(i)}$ and thus more likely that $\alpha(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*)$ is very small, and more likely the candidate point will be rejected. This can explain why for $W = 10$, the histograms have some big peaks for certain values because the algorithm got stuck there for some period of time.

Decreasing the step-width may improve this situation. When the step-width is too small, the acceptance rate $\alpha(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*)$ can be relatively big, then it is more likely to accept the candidate point. However, it can take much longer to sample all the possible values in the support of the distribution because only small jumps are possible. In some special situations, such as multiple modes well separated from each other, when the sampling keeps happening in one mode, it's very difficult to sample a candidate point from the other mode because the small step-width limits the possibility, just as what we see in Figure 2.2.

The acceptance rates (AR) for each step-width are listed in Table 2.1. AR for $W = 0.1$ is extremely high (0.932), and AR for $W = 10$ is very low (0.093), which verifies the above discussion.

This property of the MH algorithm has been recognized for a long time. Recently, new algorithms have been proposed to try to avoid some of these problems. We introduce three very recent ones in the following three sections.

Table 2.1: Acceptance rate for each scale

Scale	0.1	3	10
AR	0.932	0.194	0.039

2.2 Automatic Step Size Selection in the Metropolis-Hastings Algorithm

Although the MH algorithm is a powerful tool, its mentioned drawbacks can often create difficulties in implementation. We've seen via an example that the performance of the MH algorithm is sensitive to the choice of step-width. Graves (2005) proposes a method for automatically finding a judicious choice of step-width to use in the MH algorithm. Graves connects the problem of finding a desirable step-width with how to achieve a desirable acceptance rate in the MH algorithm. Gelman, Roberts, and Gilks (1995) give theoretical and simulation-based justification for acceptance rates of 15% to 50% yielding good performance of the MH algorithm. If so, the problem of achieving good performance of the MH algorithm by tuning the step-width reduces to obtaining a good acceptance rate by tuning the step-width. Graves (2005) empirically notes that the logit of the acceptance rate (A) is very nearly linear in the log of the step-width (W), that is,

$$\text{logit(A)} = a + b \times \log(W), \tag{2.1}$$

where $\text{logit}(y) = \log(y/(1-y))$ and suggests that model (2.1) can be used to find a good step-width for the MH algorithm via the following method.

Graves' Method.

1. Divide the burn-in period into P sections each with an equal number of iterations. For each of these use a different step-width.
2. For each iteration record a 1 if the proposal is accepted and a 0 otherwise, and thus calculate the acceptance rate A for each step-width.
3. Treat this as a dose-response experiment, with step-width taking on the role of dose and AR the role of response probability and fit the logistic regression model (2.1) to obtain $\hat{a}$ and $\hat{b}$.

4. Find the recommended step-width W^* by solving the fitted model for a certain desired acceptance rate A^*, that is,
$$W^* = \exp(\frac{\text{logit}(A^*) - \hat{a}}{\hat{b}}).$$

One key problem is how to choose a sequence of step-widths to apply in the burn-in period. Graves suggests one way. The step-widths are built in a geometric sequence with the common ratio of 2 around the initial guess of an optimal step-width, C. So once an initial guess of the optimal step-width, C, and the number of different step-widths, P, are given, the sequence of step-widths is $\{C * 2^{-\frac{P-1}{2}}, C * 2^{-\frac{P-3}{2}}, \cdots, C, \cdots, C * 2^{\frac{P-3}{2}}, C * 2^{\frac{P-1}{2}}\}$.

Based on the above construction of a sequence of step-widths, Graves (2005) performed simulations to: find an ideal logistic model; determine how accurate the initial guess of the optimal step-width needs to be; evaluate how many steps are needed and how many runs for each step are needed.

Graves' simulation study assumes that the true relationship between acceptance rate and step-width is given by $\text{logit}(A) = -5.7 - 1.12\log(W)$ and the optimal step-width W is 0.01 (yielding $A \doteq 0.368$). Three factors are used in this study. The initial guess of the optimal step-width, C, the number of the steps, P, and the number of iterations per step, M are set as follows: $C = 0.01 \times 2^k, k = \{3, 5, 7, 9, 11, 13, 15\}$, $P = \{3, 5, 7, 9, 11, 13, 15\}$, and $M = \{10, 20, 30, 40, 50\}$. For example, if C is 0.16, P is 5 and M is 20, then the sequence of the step-widths is (0.04, 0.08, 0.16, 0.32, 0.64) and each of them is run for 20 iterations. For each combination of the three factors, a logistic model was built. For over 95% of 100 simulated data sets, Graves' method yielded an acceptance rate falling between $(0.25, 0.45)$.

Graves' main observations from his simulation study are as follows. If the initial guess of the optimal step-width is exactly right, three steps and 40 trials per step are adequate. If the initial guess of the optimal step-width is not correct but too large, say too large by a factor of two, several ways can make it equally efficient: 20 each at 9 or 11 levels, 30 each at 7 levels, or 40 at 5 levels. The general rule is, the number of the steps should be large enough so that at least two step-widths smaller than the optimal are tried. If the initial guess of the optimal step-width is small, say too small by a factor or two or four, forty trials of each step are adequate. When the step-width is underestimated by a factor of eight or more, total sample sizes of 180, 220 and 280 respectively, and 20 or 40 per step size are equally effective.

2.3 Slice Sampling

Neal (2003) introduces a new MCMC method called slice sampling. The idea comes from the observation that for a univariate distribution, to sample a random variable x from $\pi(x)$, one can sample uniformly from the region under the curve of the density function and only look at the horizontal coordinate. This idea can be interpreted as follows. Suppose we wish to sample from a distribution for a variable x. The density of x is proportional to some function $f(x)$. An auxiliary variable y is introduced. A joint distribution over x and y that is defined over the region $U = \{(x, y) : 0 < y < f(x)\}$ below the curve $f(x)$ is

$$p(x,y) = \begin{cases} 1/z, & \text{if } 0 < y < f(x), \\ 0, & \text{otherwise,} \end{cases}$$

where $z = \int f(x)dx$.

So to sample x, we can sample (x, y) jointly, and then ignore y. The marginal density for x, our target distribution, is

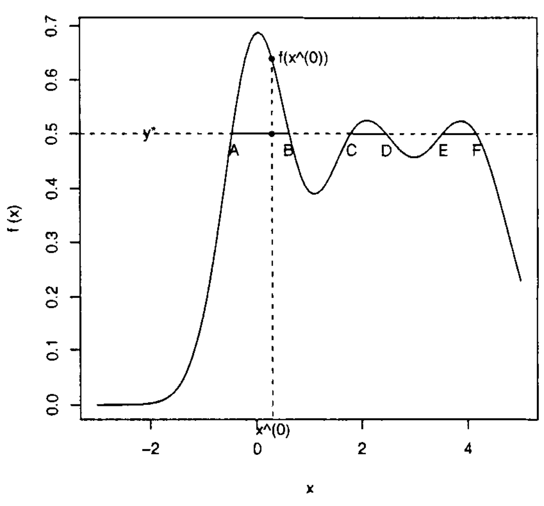
$$p(x) = \int_0^{f(x)} (1/z)dy = f(x)/z.$$

For slice sampling, the Markov chain is constructed by repeatedly alternating between the following two sampling steps. Assume the current state is $x^{(0)}$.

- Uniformly sample y on the vertical interval $(0, f(x^{(0)}))$. Call the sampled y, y^*. See Figure 2.5.
- Uniformly sample x from the set $\{x : y^* < f(x)\}$ (termed a slice), depicted in Figure 2.5 as the three horizontal lines with end-points $[A, B]$, $[C, D]$ and $[E, F]$.

The key point in slice sampling is how you can get "the horizontal interval" and sample from it to make the Markov chain converge to the invariant distribution. That is, how does one quickly sample from the set of $\{x : y^* < f(x)\}$? Neal introduces some updating strategies. We will first introduce the slice sampling method we use for univariate distribution, and then consider the multivariate case.

Figure 2.5: Slice sampling



2.3.1 Univariate slice sampling methods

Assume $f(x)$ is a function proportional to the target distribution. The current value of the variable is $x^{(0)}$ and the new state value is $x^{(1)}$. Slice sampling on a univariate distribution proceeds as follows.

Slice Sampling (Univariate).

Step 1. Draw $y \sim U(0, f(x^{(0)}))$. Define a horizontal "slice", $S = \{x : y < f(x)\}$, and $x^{(0)} \in S$.

Step 2. Find an interval $I = (L, R)$ around $x^{(0)}$ which contains much or all of the slice (discussed subsequently).

Step 3. Draw the point x_1 from the interval I (discussed subsequently).

Step 4. Repeat $1 - 3$.

To make the Markov chain converge to the right invariant distribution, the correctness of the univariate slice sampling requires the new state $x^{(1)}$ be chosen properly. This involves obtaining a random interval $I = (L, R)$ followed by selecting $x^{(1)}$ from it. There are two strategies introduced by Neal to find the interval I; the *Stepping out* procedure and the *Doubling* procedure. We will only consider the simpler of these, the Stepping out procedure.

The Stepping out procedure works as follows. Draw $y \sim U(0, f(x^{(0)}))$ and define a slice as in step 1 of Slice Sampling. Randomly position an initial interval of length E so that it contains the current state $x^{(0)}$. Repeatedly expand the interval on both ends, by increasing its length by E, until the ends satisfy $y \geq f(L)$ and $y \geq f(R)$ or the interval reaches a randomly determined maximum size.

The Stepping out procedure in detail:

Settings: Let E = estimate of the typical size of a slice and m be a parameter to control the maximum length of interval $I = (L, R)$.

Step 1. Let $L = x^{(0)} - E * U_1$ and $R = L + E$, where $U_1 \sim Unif(0,1)$.

Step 2. Let $J = \text{Floor}(mU_2)$ and $K = (m-1) - J$, where $U_2 \sim Unif(0,1)$.

Step 3. Repeat while $J > 0$ and $y < f(L)$: $L = L - E$, $J = J - 1$.

Step 4. Repeat while $K > 0$ and $y < f(R)$: $R = R + E$, $K = K + 1$.

After an interval $I = (L, R)$ has been found, the next step is to randomly draw a new point $x^{(1)}$ from this interval.

One could repeatedly uniformly sample from the interval I until a point that lies in $G = I \cap S$ is found, but this can be inefficient when G is a small portion of I. Neal introduced one method called the *Shrinkage* procedure to avoid this problem. Sample uniformly from the initial interval I; if the point drawn is not in G; shrink the interval; and repeat sampling from the new interval.

Note. In some situations, the doubling procedure may be more efficient than the stepping out procedure, because when the initial interval E is too small, the doubling procedure may expand the initial interval faster than the stepping out procedure. One main difference between the stepping out procedure and doubling procedure is, the interval I found by the stepping out procedure ensures that the set G that one should sample from is $S \cap I$, while for the doubling procedure it may turn out the set G one should sample from is only a part of $S \cap I$. This creates more complications if using the shrinkage procedure to sample a new point from the interval I found by the doubling procedure. An extra test needs to be done on the new point to determine whether it is acceptable or not, i.e. whether it is in set G. More discussion can be found in Neal (2003).

2.3.2 Multivariate slice sampling methods

Slice sampling from multivariate distributions can be extended from univariate slice sampling by applying it in each dimension in turn. However, performing the stepping out procedure or doubling procedure to search for the interval I in all p dimensions can be very time consuming. Instead one can directly sample from multiple dimensions.

Assume $\boldsymbol{x} = (x_1, \ldots, x_p)'$ is a p-dimensional variable whose density is proportional to $f(\boldsymbol{x})$. Similar to the idea of using an interval $I = (L, R)$ for the univariate case, one uses an axis-aligned hyper-rectangle $H = \{\boldsymbol{x} : L_j < x_j < R_j \text{ for } j = 1, \ldots, p\}$ containing the current point. Assume the current state is $\boldsymbol{x}^{(0)} = (x_1^{(0)}, \ldots, x_p^{(0)})'$, and the next state is $\boldsymbol{x}^{(1)} = (x_1^{(1)}, \ldots, x_p^{(1)})'$. Multivariate slice sampling is implemented as follows.

Slice sampling (Multivariate).

Step 1. Draw $y \sim U(0, f(\boldsymbol{x}^{(0)}))$. Define a slice to be $S = \{\boldsymbol{x} : y < f(\boldsymbol{x})\}$.

Step 2. Find a hyper-rectangle $H = (L_1, R_1) \times \cdots \times (L_p, R_p)$ around $\boldsymbol{x}^{(0)}$ which preferably contains at least a big part of the slice.

Step 3. Draw the point $\boldsymbol{x}^{(1)}$ from the part of the slice within this hyper-rectangle.

Step 4. Repeat Steps $1-3$.

The ideal H is the smallest hyper-rectangle containing S. But obtaining this is likely not feasible. If the variables have bounded ranges, H can be set to the whole space of the variables, but this can be very inefficient. So, in practice we have to be content with finding one hyper-rectangle that contains the current point $\boldsymbol{x}^{(0)}$. The easy way to position the hyper-rectangle H is to use a common fixed scale parameter along each axis $\boldsymbol{E} = (E_1, \ldots, E_p)$, where $E_j = E, j = 1, \ldots, p$.

Step 1. Positioning H: Along each axis, j, let the left and right bound of the hyper-rectangle be denoted L_j and R_j, respectively. H is positioned by letting $L_j = x_j^{(0)} - E * U_j$ and $R_j = L_j + E$, where $U_j \sim Unif(0, 1)$ independent for $j = 1, \ldots p$.

Step 2. Sample $\boldsymbol{x}^{(1)}$ from the rectangle H by shrinkage procedure applied to each dimension, as follows Repeat while $y > f(\boldsymbol{x}^{(1)})$: $x_j^{(1)} = L_j + U_j(R_j - L_j)$. If $x_j^{(1)} < x_j^{(0)}$, $L_j = x_j^{(1)}$, otherwise, $R_j = x_j^{(1)}$.

2.4 Equi-Energy Sampler Algorithm

The Equi-Energy (EE) sampler is proposed in Kou, Zhou, and Wong (2006). To introduce it, we begin with some definitions from statistical mechanics.

2.4.1 Problem in statistical mechanics

The distribution of a system in thermal equilibrium at temperature T is described by the Boltzmann distribution,

$$p_T(x) = \frac{1}{Z(T)} \exp(-h(x)/T), \tag{2.2}$$

where $h(x)$ is the energy function or Hamiltonian associated with the Boltzmann distribution, and $Z(T) = \sum_x \exp(-h(x)/T)$ is referred to as the partition function. For any state function $g(x)$, its expectation $\mu_g(T)$ with respect to the Boltzmann distribution is known as the Boltzmann average,

$$\mu_g(T) = \sum_x g(x) \exp(-h(x)/T)/Z(T). \tag{2.3}$$

To study the system, people are interested in estimating the Boltzmann averages $\mu_g(T)$ and the partition function $Z(T)$.

MCMC methods, for example Metropolis and MH algorithms, were applied to solve these problems, but they can perform poorly if the energy function has many local minima separated by high barriers that cannot be crossed by the proposed moves. In this situation the chain will be trapped in local energy wells and will fail to sample the Boltzmann distribution correctly. To overcome this problem, some ideas have been proposed in the literature. For example, adding auxiliary variables as in group Ising updating in Swendsen and Wang (1987) or data-augmentation in Tanner and Wong (1987). However, these ideas are problem-specific and may not work for any given problem.

Later, some dynamic Monte Carlo methods greatly improved the situation (Geyer (1991); Marinari and Parisi (1992) etc.). They were developed to simulate from the Boltzmann distribution at fixed temperatures. These methods can also be called *temperature-domain* methods. They aim to provide direct estimates of parameters such as Boltzmann averages and partition functions that are functions of temperature.

The EE algorithm of Kou, Zhou, and Wong (2006) is based on *energy domain* consideration. Energy domain methods use the **duality** between temperature domain functions and energy domain functions. They obtain the density of states and microcanonical averages (both are energy-domain functions), and then are transferred to the temperature domain to obtain the partition and the Boltzmann averages. Details are in the following section.

2.4.2 Equi-energy sampler algorithm

Before discussing the EE algorithm, we first describe some notation related to this algorithm. Assume $\pi(x)$ is the target distribution and $h(x)$ is the corresponding energy function. Then $\pi(x) \propto exp(-h(x)/T)$, which has a similar form as the Boltzmann distribution.

To apply the EE sampler algorithm, a sequence of energy levels and the associated temperatures need to be set,

$$\begin{aligned} H_0 < H_1 < \cdots < H_K < H_{K+1} = \infty, \\ 1 = T_0 < T_1 < \cdots < T_K, \end{aligned} \tag{2.4}$$

where H_0 is below the minimum energy $H_0 \leq \inf_x h(x)$. The EE sampler algorithm constructs $(K + 1)$ parallel chains. The target distribution for the k-th chain is $\pi_k(x) \propto exp(-h_k(x)/T_k)$, where $h_k(x) = \max\{h(x), H_k\}$, $k = 0, 1, \ldots, K$. Each chain corresponds to one energy level and the associated temperature. Chain $X_{[k]}$ aims to sample from $\pi_k(x)$. Chain $X_{[0]}$ aims to sample from $\pi_0(x)$ which is the target distribution $\pi(x)$. This is because

$$\pi_0(x) \propto exp(-\frac{1}{T_0} \times \max\{h(x), H_0\}), \quad \text{and} \quad T_0 = 1, \;\; h(x) \geq H_0,$$

so that

$$\pi_0(x) = \pi(x) \propto exp(-h(x)).$$

Thus, the basic idea of the EE algorithm is as follows. The distribution of the higher level chain, for example chain $X_{[k]}$, is flattened by the energy function $h_k(x)$ and the associated temperature level T_k, which avoids the problem of local traps. Once the higher level chain is constructed, in the process of the construction of the lower level chain, the new point can "jump" to a new x with similar energy level (i.e. $y = h(x)$) to the current point with the help of the higher-level chain. We call this an Equi-energy jump (EE jump). That is, the new point is randomly chosen from similar energy sets from the higher-level chain. An EE jump helps to avoid the local traps of the lower-level chain. The last chain, $X_{[0]}$, is proved to have steady-state distribution $\pi_0 = \pi$ in Kou, Zhou, and Wong (2006). We call the empirical equi-energy sets energy rings.

To construct the energy rings, the state space Ψ is partitioned according to the energy levels, $\Psi = \bigcup_{j=0}^{K} D_j$, where $D_j = \{x : h(x) \in [H_j, H_{j+1}]\}$ for $0 \le j \le K$, are the energy sets determined by the energy sequence (2.4). For any $x \in \Psi$, let $I(x)$ denote the partition index such that $I(x) = j$, if $x \in D_j$, i.e. if $h(x) \in [H_j, H_{j+1})$.

The EE algorithm starts by constructing the highest level chain $X_{[K]}$, then moves on to the the lower level chains, and ends with the lowest level chain $X_{[0]}$. Each chain has its own empirical energy rings so that $D_j^{[k]}$ denotes the j-th energy ring of the k-th chain, for $j = 0, \ldots, K$ and $k = 0, \ldots, K$. The two basic types of move for generating a new point in the EE algorithm are: (i) an MH local move. (ii) an EE jump. The algorithm is described in the order of the chains being constructed.

EE algorithm.

Step 1. The first chain $X_{[K]}$ is constructed by the MH algorithm. After a burn-in period B, the generated points are assigned into different "energy rings" $D_j^{[K]}$ where $j = I(X_{\{[K],i\}})$ and i denotes the i-th step of the chain. Note that the points can belong to any energy ring.

Step 2. The second parallel chain $X_{[K-1]}$ starts after the first chain $X_{[K]}$ has run $(B + N)$ steps. Simultaneously, the first chain still keeps running and its generated points are continually assigned into energy rings. The second chain $X_{[K-1]}$ is constructed either by an MH move or by an EE jump. The first point, $X_{\{[K-1],1\}}$, is generated from an MH step. The point $X_{\{[K-1],i+1\}}, i \ge 1$, is also generated by an MH move, if the energy ring $D_j^{[K]}$ of chain $X_{[K]}$ is empty, where j is the energy level of the current point $X_{\{[K-1],i\}}$. When the energy ring $D_j^{[K]}$ is not empty, the next point $X_{\{[K-1],i+1\}}$ is generated via an MH local move with probability J, or through an EE-jump with probability $1 - J$. If the EE-jump is implemented, a state x^* is uniformly chosen from the energy ring $D_j^{[K]}$, and the chosen x^* is accepted as $X_{\{[K-1],i+1\}}$ with probability

$$\min\left\{1, \frac{\pi_{K-1}(x^*)\pi_K(X_{\{[K-1],i\}})}{\pi_{K-1}(X_{\{[K-1],i\}})\pi_K(x^*)}\right\}.$$

If x^* is rejected, $X_{\{[K-1],i+1\}}$ keeps the old value $X_{\{[K-1],i\}}$.

Similar to with the first chain, after an initial burn-in period of B steps, the generated points of the second chain are assigned into different "energy rings" $D_j^{[K-1]}$ where $j = I(X_{\{[K-1],i\}})$. Note by this time the first chain $X_{[K]}$ has run $(2B + N)$ steps.

$\vdots$

Step $(K - i + 1)$. The $(K - k + 1)$-th parallel chain $X_{[k]}$ starts after chain $X_{[k-1]}$ has run $(B + N)$ steps. All the previous chains are continually updated. The construction of chain $X_{[k]}$ is similar to that of the second chain $X_{[K-1]}$. The point $X_{\{[k],i+1\}}$ is generated by an MH step when: (i) $i = 0$; (ii) $D_j^{(k+1)} = \emptyset$; (iii) with probability J if $D_j^{[k+1]} \neq \emptyset$; or by an EE-jump with probability $1 - J$ when $D_j^{[k+1]} \neq \emptyset$. x^* is chosen from $D_j^{[k+1]}$ and the acceptance rate is

$$\min\left\{1, \frac{\pi_k(x^*)\pi_{k+1}(X_{\{[k],i\}})}{\pi_k(X_{\{[k],i\}})\pi_{k+1}(x^*)}\right\},$$

where $k = 0, 1, \ldots, K$. After the initial period of B steps, the generated points are assigned into "energy rings" $D_j^{[k]}$, where $j = I(X_{\{[k],i\}})$.

$\vdots$

Step $(K + 1)$. The last chain $X_{[0]}$ is constructed and has target distribution $\pi(x)$.

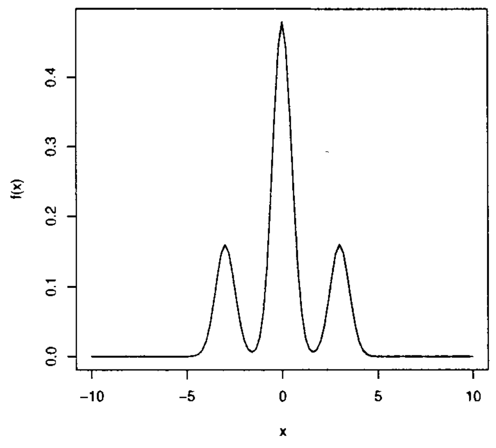
In the implementation of the EE algorithm, Kou, Zhou, and Wong (2006) suggest some practical settings for the parameters. Given the lowest and highest energy levels H_0 and H_K, the other energy levels can be set as a geometric progression, that is, setting $\log(H_{j+1} - H_j)$ evenly spaced. Choosing the temperature such that $(H_{j+1} - H_j)/T_j \approx O$, with $O \in [1, 5]$, often works well. The choice of K, the number of temperature and energy levels, depends on the complexity of the problem. Usually more chains and energy levels are needed if the target distribution is high-dimensional and multi-modal. The authors recommend K be chosen roughly proportional to the dimensionality of the target distribution. The EE jump probability J should be chosen between 0.05 and 0.30. We illustrate the EE algorithm via the following example.

2.4.3 Example

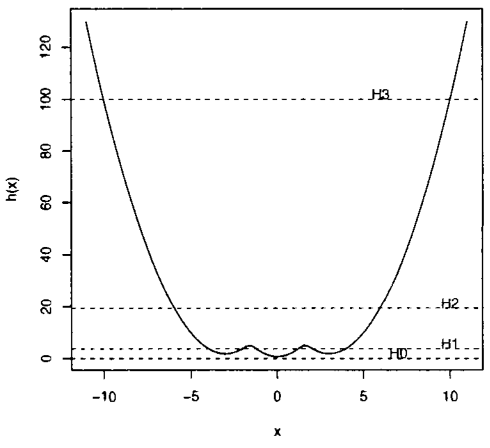
Assume the target distribution is a mixture of univariate normal distributions,

$$f(x) = \sum_{l=1}^{m} \frac{w_l}{\sqrt{2\pi}\sigma_l} \exp\left\{\frac{1}{2\sigma_l^2}(x - \mu_l)^2\right\},$$

Figure 2.6: EE energy levels



(a) f(x) plot

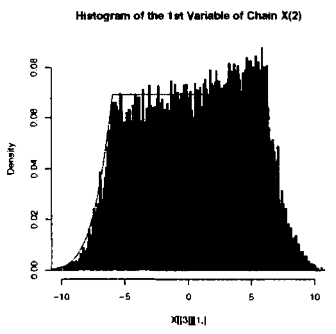


(b) Energy function h(x) plot with the energy levels

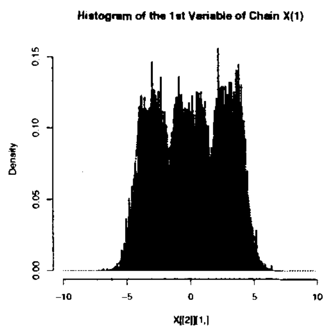
where $m = 3$ is the number of normal distributions, $\sigma_l = 0.5$ is the common standard deviation of the l-th normal distribution, w_l are the weights assigned to each distribution, $w_1 = 1/5$, $w_2 = 3/5$ and $w_3 = 1/5$. And μ_l are the means, $\mu_1 = -3$, $\mu_2 = 0$ and $\mu_3 = 3$. Set the number of energy levels to $K = 2$, constant $O = 4$ and the probability of an EE-jump to $J = 0.2$. The step-width of the MH algorithm is set to 2.6. The simulation starts from the initial point $x^{(0)} = -3$. Since $\inf h(x) = 0.737$, we set the lowest energy level $H_0 = 0$. The highest energy level H_3 is set as 100, that corresponds to $f(x) = 3.720076e - 44$ which is almost 0. So the rest of the energy levels are set in a geometric progression. The energy level 1, H_1, is set to $\inf h(x) + 3 = 3.74$, and the energy level 2, $H_2 = H_1 * (H_3/H_1)^{1/2}$. The temperature levels except T_0 are set using the relationship with the energy level $(H_{j+1} - H_j)/T_j \approx O$, where $O = 4$ here. So, the temperature levels are 1, 3.9 and 20.17.

Figure 2.6(a) is a plot of the target distribution. Figure 2.6(b) is a plot of the energy function, $h(x)$, with the energy levels H_0, H_1, H_2 and H_3 depicted as horizontal dotted lines. Figure 2.7 gives histograms of the samples from the three chains in the EE algorithm together with the target distributions. Figure 2.7(a) is the first chain $X_{[2]}$, which corresponds to the target distribution $f_2(x) \propto exp(-h_2(x)/T_2)$. Figure 2.7(b) is the second chain $X_{[1]}$,

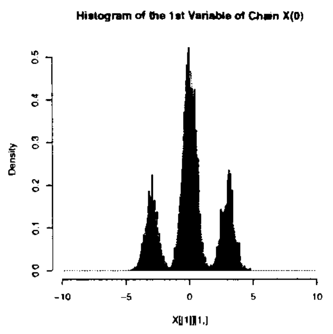
Figure 2.7: EE algorithm output



(a) Chain $X_{[2]}$



(b) Chain $X_{[1]}$



(c) Chain $X_{[0]}$

which corresponds to the target distribution $f_1(x) \propto exp(-h_1(x)/T_1)$. Finally, Figure 2.7(c) is the last chain $X_{[0]}$, which corresponds to the target distribution $f(x)$.

Chapter 3

Screening Experiment

3.1 Introduction

In the previous chapter, we reviewed a number of sampling algorithms. In this chapter, a screening experiment will be performed to study and compare the algorithms and to explore the parameters that have a significant effect on the algorithm performance. In this screening experiment, we first define the parameters related to the target distribution and to the algorithms that we feel may impact the performance. Then we set the levels of these parameters. A 36-run experimental design is used in the simulation. The simulation is done repeatedly for each algorithm to get five replicates. A model selection procedure is applied to the average of the five replicate data to find the parameters that most affect the algorithm performance.

After finding the important parameters, we try to choose the best settings of the algorithm parameters, to maximize the overall performance of each method. We will use robust parameter design to find the best or better settings of the parameters.

The five algorithms considered in the simulation study are:

MH algorithm. The MH algorithm with a uniform proposal distribution.

Graves' method. This method models *Acceptance Rate* as a function of *Step-width* in the burn-in period of the MH algorithm, then uses the recommended step-width in the MH algorithm after the burn-in.

Slice sampling algorithm. Slice sampling (Multivariate) is used, since the target distribution is a bivariate distribution (see Section 3.2.1).

EE-MH algorithm. The Equi-energy sampling algorithm with an MH algorithm local move.

EE-Slice algorithm. The Equi-energy sampling algorithm with a Slice sampling local move.

3.2 Factors and Levels

Because the five algorithms do not all have the same tuning parameters, the experiment is carried out as five separate screening experiments. Below, we discuss all of the factors and levels included in the study, beginning with the target distributions (which are common to all five methods), and then move on to the algorithm-specific parameters. Each of the parameters discussed will be considered at two-levels or three-levels in the study.

3.2.1 The target distribution parameters

To form various possible distributional forms, we use a mixture of bivariate normal distributions $f(\boldsymbol{x})$ (as in the example of Chapter 2),

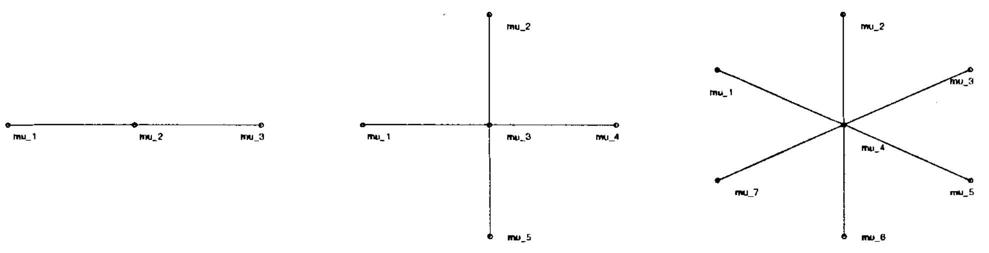
$$f(\boldsymbol{x}) = \sum_{l=1}^{m} \frac{w_l}{2\pi\sigma_l^2} exp\left\{-\frac{1}{2\sigma_l^2}(\boldsymbol{x}-\boldsymbol{\mu}_l)'(\boldsymbol{x}-\boldsymbol{\mu}_l)\right\},$$

where $\sigma_l = 0.5$ is the standard deviation for each component of each bivariate normal distribution, m is the number of bivariate normal distributions, w_l are the weights assigned to each distribution, and $\boldsymbol{\mu}_l = (\mu_{1l}, \mu_{2l})'$ are the mean vectors. The problem of generating different distributional shapes depends strongly on the number of mixture components, m, the relative locations of $\boldsymbol{\mu}_l$, and their weights,w_l. The positions of the mean vectors are controlled by the distance, d, between $\boldsymbol{\mu}_l$ along a cross shaped grid (termed grid spacing).

Based on observations of the MH algorithm example in Chapter 2 and a number of related examples, we had some general idea of what shapes of distribution impacts the performance of the MH algorithm, as well as the performances of other algorithms. For example, distributions with several modes, very separated, partly connected, or barely separated; distributions with modes that are of equal size, or a mixture of unequal sizes–one single main peak surrounded by several small modes. So, by changing the values of the parameters m, w_l and d, we can control the modes of the distribution in terms of number, relative heights, and locations, thus generating distributions with varying desired aspects.

We set m, the number of the bivariate normal distributions, at three levels; 3, 5, 7. Each level also represents a certain pattern of the mean vector positions as depicted in Figure 3.1:

Figure 3.1: Positions of the bivariate distributions



(a) m=3 (b) m=5 (c) m=7

- $m = 3$. Three mean vectors are along one line ($\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\mu}_3$) as in Figure 3.1(a).
- $m = 5$. The five mean vectors are positioned at the ends ($\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\mu}_4$, $\boldsymbol{\mu}_5$) and the center $\boldsymbol{\mu}_3$ of the cross as in Figure 3.1(b).
- $m = 7$. The two adjacent lines are at 60 degree angles to each other. The seven mean vectors are positioned at the ends ($\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\mu}_3$, $\boldsymbol{\mu}_5$, $\boldsymbol{\mu}_6$, $\boldsymbol{\mu}_7$) and the center, $\boldsymbol{\mu}_4$, of the lines depicted in Figure 3.1(c).

The grid spacing, d, the distances between the mean vectors along the lines in Figure 3.1, are set at three different levels; 1, 3, 5. The reason for choosing these particular values is, with $\sigma_i = 0.5$, they produce modes that are barely separate; partly connected; and far apart, respectively. The third parameter, weight $\boldsymbol{w} = (w_1, \cdots, w_m)'$, is considered at two levels: $w_1 = \cdots = w_m = 1/m$, making the modes equally sized; and $w_{(m+1)/2} = 2/m$ and $w_i = (m-2)/[m(m-1)], i = \{1, \cdots, (m-1)/2, (m+3)/2, \cdots, m\}$, so as to make one large central peak surrounded by smaller modes. Table 3.1 is a summary of the parameter settings related to the target distribution.

Table 3.1: Target distribution parameter settings

	Level 0	Level 1	Level 2	OA col
Grid spacing (d)	1	3	5	2
Normal dist. No. (m)	3	5	7	3
Weight ($\boldsymbol{w}$)	$(\frac{1}{m}, \cdots, \frac{1}{m})'$	$\frac{1}{m}(\frac{m-2}{m-1}, \cdots, 2, \frac{m-2}{m-1}, \cdots)'$		10

3.2.2 Algorithm Parameter Settings

In the screening experiment, the algorithms share three common tuning parameters: initial value $(V_1, V_2)'$, number of iterations (N), and number of interactions in the burn-in period (B). In addition, each algorithm has its own specific parameters. We first introduce the parameter settings of the common algorithm parameters, and then the algorithm specific parameters for the algorithms in turn. A summary of the parameter settings is given in Table 3.2.

Common Algorithm Parameters

The initial values $(V_1, V_2)'$ corresponding to the variate $\boldsymbol{x} = (x_1, x_2)'$ are both set equal to V. They are considered at two levels; $-d$ and 0 (the weighted mean of the mean vectors, $\sum_{i=1}^{m} w_i \mu_{ij} = 0$, $j = 1, 2$). We consider the number of iterations, N, equal to 10000 and 40000, since $N = 10000$ usually works well for the MH algorithm and slice sampling, but EE sampling needs a larger N. For similar reasons, we set the burn-in period B at 500 and 2000.

Algorithm Specific Parameters

MH algorithm

Our implementation of the MH algorithm uses the uniform distribution to generate the candidate points for each variate. Let $(W_1, W_2)'$ denote the vector of the step-widths of the proposal distribution for MH algorithm, where W_j, is the step-width of the uniform distribution for variate x_j, $j = 1, 2$. That is, the candidate point for variate is generated through a uniform distribution, $Unif(x_j^{(k)} - W_j, x_j^{(k)} + W_j), j = 1, 2$, where $(x_1^{(k)}, x_2^{(k)})$ is the current point (iteration k). In our study we set $W_1 = W_2 = W$. Since the grid spacing d is 1, 3, and 5, we choose three values for W; 0.1, 2.6, and 5.1. In this way we consider cases where the step-width is much smaller, similar to and much bigger than the grid spacing. We anticipate that a small step-width relative to grid spacing will impact the ability of the algorithm to adequately explore the support of the target distribution.

Slice Sampling Algorithm

The strategy of slice sampling a bivariate target distribution is sampling directly from multiple dimensions. Let $(E_1, E_2)'$ denote the vector of scale parameters to position the hyper-rectangle H that contains the current point $(x_1^{(k)}, x_2^{(k)})$. We let $E_1 = E_2 = E$. As with the MH algorithm, E is set to have the levels (0.1, 2.6, 5.1), to match the three levels of grid spacing. Thus E is playing a similar role for slice sampling as W does for MH.

Graves' Method

In the simulation study, we would like to see whether using model (2.1) will find a step-width that can improve the performance of the MH algorithm or not. We fit the model in the burn-in period of the MH algorithm. A sequence of P step-widths are constructed, and each step-width is run for B/P iterations with their acceptance rates recorded. These step-widths and acceptance rates are used to fit model (2.1). A step-width will be recommended through the model by an ideal acceptance rate, and then be applied in the MH algorithm after the burn-in period.

Based on our observations, a reasonable step-width is around $1.5d$. We want the sequence of step-widths to cover this value, and also to cover some values that are smaller and bigger than this value. We thus construct the step-width sequence as follows: we consider the center step-width C at three levels, 1, 2, and 3, and P at two levels, 5 and 11, and generate a sequence of step-widths (as suggested in Graves (2005)) by

$$\{C * 2^{-\frac{P-1}{2}}, C * 2^{-\frac{P-3}{2}}, \cdots, C, \cdots, C * 2^{\frac{P-3}{2}}, C * 2^{\frac{P-1}{2}}\}.$$

The last parameter for Graves' method is the ideal acceptance rate, A, which is used via inverting model (2.1) to find the recommended step-width. Graves suggests the ideal acceptance rate be between 0.25 and 0.45. We consider three levels; 0.2, 0.4, and 0.6.

EE-MH Algorithm

The EE-MH algorithm is the EE sampling algorithm with an MH sampling local move. It has five parameters: the number of energy levels (K), energy level 1 (H_1), constant (O)

related to the energy and temperature levels, EE jump probability (J), and the step-width (W).

The following parameter settings consider the authors' suggestions in the original EE sampling paper, Kou, Zhou, and Wong (2006). Based on preliminary investigations, we set K to two levels; 2 and 4. The energy and temperature levels are set as follows: assume the energy levels are $H_i, i = 0, \cdots, K+1$ and the temperature levels are $T_i, i = 0, \cdots, K$. Since the lowest energy level H_0 must be $\leq \inf_x h(x)$, and in our case $\inf_x h(x) = \inf_x(-\log(f(x)))$ is a small positive number, H_0 is set to 0. The highest energy level, H_{K+1}, which corresponds to $f(x)$ close to 0, is set to 100. The rest of the energy levels are set in a geometric progression. So the starting point of the geometric progression H_1 is needed to construct the sequence. Once the starting point of the geometric progression, H_1, is determined, the other energy levels can be set as

$$H_2 = H_1 \times (\frac{H_{K+1}}{H_2})^{\frac{1}{K}}, \cdots, H_K = H_{K-1} \times (\frac{H_{K+1}}{H_2})^{\frac{1}{K}}.$$

We consider three levels of H_1; $\inf_x h(x)+1$, $\inf_x h(x)+3$, and $\inf_x h(x)+5$. With the energy levels set, the temperature levels can be obtained by $T_i = (H_{i+1} - H_i)/O, i = 1, \cdots, K$, where $T_0 = 1$, and O is a constant suggested in Kou, Zhou, and Wong (2006) between 1 and 5. We consider O at two levels; 2 and 4. The fourth parameter, J, is the probability of an EE-jump, and is suggested in Kou, Zhou, and Wong (2006) to be set between 0.05 and 0.30. We consider three levels, 0.10, 0.20, and 0.30. For the final parameter, step-width (W), we use the same values as in the MH algorithm.

EE-Slice Algorithm

The EE-Slice algorithm is the EE sampling algorithm with a Slice sampling local move. It has five parameters: the number of energy levels (K), energy level 1 (H_1), constant (O) related to the energy and temperature levels, EE-jump probability (J), and the scale estimate (E). The first four parameters are considered at the same levels as those in EE-MH algorithm, and the scale parameter, E, is considered at the same levels as in the Slice sampling algorithm.

Table 3.2: Algorithm parameter settings

		Common algorithm parameters			
		Level 0	Level 1	Level 2	OA col
Initial value (V)		$-d$	0		11
Iteration No. (N)		10000	40000		12
Burn-in (B)		500	2000		13
		Algorithm specific parameters			
		Level 0	Level 1	Level 2	OA col
MH	Step-width (W)	0.1	2.6	5.1	4
Graves'	No. of points (P)	5	11		14
	Initial guess of the optimal step-width (C)	1	2	3	5
	Acceptance rate (A)	0.2	0.4	0.6	6
Slice	Scale parameter (E)	0.1	2.6	5.1	4
EE-MH	Step-width (W)	0.1	2.6	5.1	4
	Energy level No. (K)	2	4		14
	EE jump prob. (J)	0.1	0.2	0.3	5
	Energy level 1 (H_1)	1	3	5	6
	Constant (O)	2	4		15
EE-Slice	Scale parameter (E)	0.1	2.6	5.1	4
	Energy level No. (K)	2	4		14
	EE jump prob.(J)	0.1	0.2	0.3	5
	Energy level 1 (H_1)	1	3	5	6
	Constant (O)	2	4		15

3.3 Screening Experiment

3.3.1 Design matrix

We use the 36-run orthogonal array (OA) in Zhang, Pang, and Wang (2001) as the basis for the design matrices for our screening experiments. This is a $6^1 \times 3^8 \times 2^{10}$ design (see Table 3.3). That is, it has one 6-level columns, 8 3-level columns and 10 2-level columns. From the summary of the five algorithms, we see that the MH and Slice sampling algorithms have 3 three-level factors and 4 two-level factors, Graves' method has 4 three-level factors and 5 two-level factors, while both the EE-MH and EE-Slice algorithms have 5 three-level factors and 6 two-level factors. So for each algorithm we pick out certain columns from the base OA, and use them as the settings of the parameters of that algorithm. We run five independent experiments, one for each algorithm. For the common parameters, including the distribution parameters and the common algorithm parameters, all of the algorithms use the same settings. The settings of the distribution parameters d, m and w are done via columns 2, 3 and 10, respectively. The settings of the common algorithm parameters V, N and B are done via columns 11, 12 and 13, respectively. The settings of the algorithm specific parameters are as follows: The step-width of MH and the scale parameter of Slice sampling are set using column 4. For Graves' method, the number of points, P, the initial guess of the optimal step-width, C, and the acceptance rate, A, are set using columns 14, 5 and 6, respectively. For the two EE algorithms, the number of energy levels, K, the EE-jump probability, J, energy level 1, H_1, and constant, O, are set using columns 14, 5, 6, and 15,respectively. The step-width and the scale parameter of the EE algorithms are set using column 4. The above information is summarized in Tables 3.1 and 3.2. The design matrix of each algorithm is given in Appendix A, the first column of y-responses in Tables A.1-A.5.

3.3.2 Response to measure the goodness of fit

For each algorithm, a 36-run simulation is carried out. Each run of the simulation is one application of the parameter settings of the algorithm based on the algorithm's corresponding design matrix. The output of each run from one algorithm is a sample that is purportedly from the target distribution. An appropriate goodness-of-fit measure is needed to evaluate

Table 3.3: Orthogonal Array OA

col	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	0	1	1	2	2	1	1	2	2	0	0	0	0	0	0	0	0	0	0
2	1	1	1	2	2	2	2	1	1	1	1	0	0	0	1	0	1	0	1
3	1	2	2	1	1	1	1	2	2	0	0	1	1	1	1	0	0	0	1
4	0	2	2	1	1	2	2	1	1	1	1	1	1	0	1	0	0	1	0
5	4	1	2	1	2	2	0	2	0	0	0	1	0	0	1	1	1	1	1
6	5	1	2	1	2	0	2	0	2	1	0	0	1	0	0	1	0	1	1
7	5	2	1	2	1	2	0	2	0	0	1	0	0	1	1	1	0	1	0
8	4	2	1	2	1	0	2	0	2	1	1	1	0	1	0	1	0	0	1
9	2	1	2	2	1	0	1	1	0	0	1	0	1	1	0	0	1	1	1
10	3	1	2	2	1	1	0	0	1	1	0	1	0	1	0	0	1	1	0
11	3	2	1	1	2	0	1	1	0	0	1	1	1	0	0	1	1	0	0
12	2	2	1	1	2	1	0	0	1	1	0	0	1	1	1	1	1	0	0
13	2	2	2	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0
14	3	2	2	0	0	0	0	2	2	1	1	0	0	0	1	0	1	0	1
15	3	0	0	2	2	2	2	0	0	0	0	1	1	1	1	0	0	0	1
16	2	0	0	2	2	0	0	2	2	1	1	1	1	0	1	0	0	1	0
17	0	2	0	2	0	0	1	0	1	0	0	1	0	0	1	1	1	1	1
18	1	2	0	2	0	1	0	1	0	1	0	0	1	0	0	1	0	1	1
19	1	0	2	0	2	0	1	0	1	0	1	0	0	1	1	1	0	1	0
20	0	0	2	0	2	1	0	1	0	1	1	1	0	1	0	1	0	0	1
21	4	2	0	0	2	1	2	2	1	0	1	0	1	1	0	0	1	1	1
22	5	2	0	0	2	2	1	1	2	1	0	1	0	1	0	0	1	1	0
23	5	0	2	2	0	1	2	2	1	0	1	1	1	0	0	1	1	0	0
24	4	0	2	2	0	2	1	1	2	1	0	0	1	1	1	1	1	0	0
25	4	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0
26	5	0	0	1	1	1	1	0	0	1	1	0	0	0	1	0	1	0	1
27	5	1	1	0	0	0	0	1	1	0	0	1	1	1	1	0	0	0	1
28	4	1	1	0	0	1	1	0	0	1	1	1	1	0	1	0	0	1	0
29	2	0	1	0	1	1	2	1	2	0	0	1	0	0	1	1	1	1	1
30	3	0	1	0	1	2	1	2	1	1	0	0	1	0	0	1	0	1	1
31	3	1	0	1	0	1	2	1	2	0	1	0	0	1	1	1	0	1	0
32	2	1	0	1	0	2	1	2	1	1	1	1	0	1	0	1	0	0	1
33	0	0	1	1	0	2	0	0	2	0	1	0	1	1	0	0	1	1	1
34	1	0	1	1	0	0	2	2	0	1	0	1	0	1	0	0	1	1	0
35	1	1	0	0	1	2	0	0	2	0	1	1	1	0	0	1	1	0	0
36	0	1	0	0	1	0	2	2	0	1	0	0	1	1	1	1	1	0	0

how closely the algorithm output matches the true target distribution. The Average Absolute Distance (AAD) was used for this purpose. It measures the "average absolute distance" between the true cumulative distribution function (CDF) of the target distribution and the empirical cumulative distribution function (ECDF) from the sample.

We define AAD as follows. Assume $F(\boldsymbol{x}) = \int_{-\infty}^{x_1} \int_{-\infty}^{x_2} f(t_1, t_2) dt_1 dt_2$ is the CDF of the of the target distribution, $S(\boldsymbol{x})$ =*the proportion of* (x_{1i}, x_{2i}) *satisfying* $x_{1i} \leq x_1$ *and* $x_{2i} \leq x_2$ is the ECDF from the sample. The average absolute distance (AAD) is defined as

$$AAD = \frac{\sum_{i=1}^{N} |S(\boldsymbol{x}^{(i)}) - F(\boldsymbol{x}^{(i)})|}{N},$$

where N is the number of the generated points. We use AAD as the response, y, for each simulation run in our experiments.

3.3.3 Analysis based on one replicate

We ran the simulation once for the parameters of the five algorithms set based on their design matrix, and got 36 AAD's corresponding to different parameter settings for each algorithm. The results of the five experiments are summarized in Tables A.1-A.5 of Appendix A. A very rough preliminary look at the data is given in the first two columns of Table 3.4. The design is ignored and the mean and standard deviation over the 36 runs is given for each algorithm.

Table 3.4: Response comparison

	One replicate		Five replicates	
Algorithm	$\bar{y}$	sd_y	$\bar{y}$	sd_y
MH	0.0780	0.0530	0.0768	0.0470
Slice	0.0805	0.0604	0.0836	0.0536
Graves	0.0854	0.0524	0.0881	0.0536
EEMH	0.0623	0.0346	0.0658	0.0382
EESlice	0.0682	0.0489	0.0716	0.0475

The table suggests that EE-MH has the smallest average and variance of AAD, which shows it potentially has the best goodness of fit and stability. The EE-Slice algorithm also seems to perform well. Slice sampling and Graves' method do not seem to perform better than the MH algorithm.

Table 3.5: Model selection criteria

Criteria	Formula	Note
Akaike Information Criterion	$AIC = -2logL + 2p$	L is the likelihood, p is the No. of predictors of the model plus 1. The smaller the AIC value, the better the model.
Bayesian Information Criterion	$BIC = -2logL + log(n) \times p$	n is the No. of observations. The smaller the BIC value, the better the model.
R-square	$Rsq = 1 - \frac{RSS_p}{RSS_1}$	RSS_p is the residual sum squares, RSS_1 is the total sum square of the values of the y-variable subtracting the mean. The bigger the Rsq value, the better the model.
Adjusted R-square	$Adjr2 = 1 - (1 - Rsq)\frac{n-1}{n-p}$	The bigger the $Adjr2$ value, the better the model.

In the screening experiment, we are interested in what parameters play important roles in each algorithm. Model selection will be applied to find the best subsets among the main effects (linear and quadratic) and their two-way interactions. The main effects considered include the two-level parameters, the linear term of three-level parameters and the quadratic term of three-level parameters. We use an all subsets regression to find all subsets up to size 6, restricting to weak-heredity models (i.e. models for which at least one of the factors in every two-factor interaction is also in the model), and rank them by the Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Adjusted R-square ($Adjr2$) and R-square (Rsq). We also looked at the criteria values for subset size 7, but since the criteria values of size 6 are very close to those of size 7, we stopped the exploration on subset size 6. Table 3.5 is a summary of the formulae for and some notes on these criteria. More details about these criteria can be found in Miller (2002). We looked at the best subsets of size and compared to those with 6 effects and found modest improvement. Thus, we decided to look at the best model of size 6.

Before looking at the subset search results, we review the main effects of each algorithm.

MH algorithm

d_L	Linear term of the grid spacing
d_Q	Quadratic term of the grid spacing
m_L	Linear term of the number of the bivariate normal distributions
m_Q	Quadratic term of the number of the bivariate normal distributions
w	Weight
V	Initial value
N	Iteration number
B	Burn-in number
W_L	Linear term of step-width
W_Q	Quadratic term of step-width

Slice sampling algorithm

d_L	Linear term of the grid spacing
d_Q	Quadratic term of the grid spacing
m_L	Linear term of the number of the bivariate normal distributions
m_Q	Quadratic term of the number of the bivariate normal distributions
w	Weight

V	Initial value
N	Iteration number
B	Burn-in number
E_L	Linear term of scale estimate
E_Q	Quadratic term of scale estimate

Graves method

d_L	Linear term of the grid spacing
d_Q	Quadratic term of the grid spacing
m_L	Linear term of the number of the bivariate normal distributions
m_Q	Quadratic term of the number of the bivariate normal distributions
w	Weight
V	Initial value
N	Iteration number
B	Burn-in number
P	Number of the step-width in the sequence
C_L	Linear term of the initial guess of the optimal step-width
C_Q	Quadratic term of the initial guess of the optimal step-width
A_L	Linear term of the acceptance rate
A_Q	Quadratic term of the acceptance rate

EE-MH

d_L	Linear term of the grid spacing
d_Q	Quadratic term of the grid spacing
m_L	Linear term of the number of the bivariate normal distributions
m_Q	Quadratic term of the number of the bivariate normal distributions
w	Weight
V	Initial value
N	Iteration number
B	Burn-in number
W_L	Linear term of step-width
W_Q	Quadratic term of step-width
K	Energy levels

J_L	Linear term of EE jump probability
J_Q	Quadratic term of EE jump probability
H_{1L}	Linear term of 1st energy level
H_{1Q}	Quadratic term of 1st energy level
O	Constant

EE-Slice

d_L	Linear term of the grid spacing
d_Q	Quadratic term of the grid spacing
m_L	Linear term of the number of the bivariate normal distributions
m_Q	Quadratic term of the number of the bivariate normal distributions
w	Weight
V	Initial value
N	Iteration number
B	Burn-in number
E_L	Linear term of scale estimate
E_Q	Quadratic term of scale estimate
K	Energy levels
J_L	Linear term of EE jump probability
J_Q	Quadratic term of EE jump probability
H_{1L}	Linear term of 1st energy level
H_{1Q}	Quadratic term of 1st energy level
O	Constant

We use all subsets regression to find all the subsets and picked the best two weak hereditary subsets of different size based on the residual sum square RSS_p. Table 3.6 shows the top 2 weak hereditary subsets up to size 6. The best subset of size 6 for the MH algorithm is: d_L, d_Q, W_L, $d_L B$, $d_Q W_L$, and $d_Q m_Q$. So the important parameters are d, m, W and B. The best subset of size 6 for the slice sampling algorithm is: d_L, m_Q, E_L, $d_L E_L$, $d_Q m_Q$, $m_Q E_L$. The important parameters are d, m and E. The best subset of size 6 for Graves' method is: d_L, d_Q, V, A_L, $d_Q m_L$ and VA_L. The important parameters are d, m, V and A. The best subset of size 6 for the EE-MH algorithm is: N, W_L, J_L, $d_L J_L$, $d_Q J_L$ and $W_L H_{1L}$. The important parameters are d, N, W, J and H_1. The best subset of size 6 for the EE-Slice algorithm is: m_L, w, E_L, E_Q, d_LE_L, wE_L, The important parameters are d, m, w and E.

3.3.4 Analysis based on five replicates

The above analysis is based on one replicate of the 36 run simulation. We repeated each experiment simulation a second time and found there to be some significant differences from the first experiment. For example, Table 3.7 shows the percent differences of AAD for the 2nd replicate of MH compared with the 1st replicate. For some runs the difference is fairly high. Run 12 and 36 both have over 100% difference. Another eleven runs have over 15% percent differences. This tells us the simulation performances of the MH algorithm are not very stable across the replicates. A similar result was also seen with the other algorithms. So we repeated the experiments another three times and observed the algorithms' behavior based on the average response of the five replicates. A very rough preliminary look at the five algorithms on the average AAD and the standard deviation of each run is given in the last two columns of Table 3.4. The EE-MH algorithm has the smallest average and standard deviation. EE-Slice has small average too but the standard deviation is a bit worse. Slice sampling and Graves' method do not seem to perform better than the MH algorithm.

Model selection for the mean of the replicates

All subsets regression is applied to the average AAD, $\bar{y}$. Table 3.8 displays the top two weak hereditary subsets up to size 6 of the five algorithms. The best subset of size 6 for the MH algorithm is d_L, d_Q, W_L, d_LW_L, d_Lm_Q, VW_L. The important parameters are d, m, W and V. The best subset of size 6 for Slice sampling algorithm is d_L, d_Q, E_L, d_LE_L, d_QE_L, d_QE_Q. The important parameters are d and E. The best subset of size 6 for Graves' method is d_L, d_Q, A_L, d_Lm_L, d_LV, d_LA_L. The important parameters are d, m, A and V. The best subset of size 6 for the EE-MH algorithm is W_L, W_Q, d_LW_L, d_LW_Q, m_LW_L, W_LO. The important parameters are d, m, W and O. The best subset of size 6 for the EE-Slice algorithm is m_L, N, E_L, E_Q, m_Lw, NE_L. The important parameters are m, w, N and E.

Comparing with the model selection results of one replicate, most of the significant factors of one replicate are also significant in the subsets of five replicate. But there is also some difference between these two cases. For example, for the MH algorithm, burn-in B is significant based on one replicate, but not on the average of five replicates. Considering the

Table 3.6: Top 2 weak hereditary subsets up to size 6 on response of one replicate

	Size	Selected factors						AIC	BIC	$Adjr2$	Rsq
	1	W_L						−228.080	−224.913	0.403	0.420
	1	d_L						−214.281	−211.114	0.124	0.149
	2	d_L	W_L					−236.741	−231.991	0.542	0.568
	2	W_L	d_LW_L					−236.109	−231.358	0.534	0.561
	3	d_L	W_L	d_LW_L				−248.994	−242.660	0.682	0.710
M	3	d_L	W_L	d_LB				−241.732	−235.398	0.611	0.645
H	4	d_L	W_L	E	d_LW_L			−253.393	−245.475	0.725	0.757
	4	d_L	W_L	d_LW_L	d_QW_L			−251.835	−243.918	0.713	0.746
	5	d_L	m_Q	W_L	d_LW_L	d_Qm_Q		−259.090	−249.589	0.771	0.804
	5	d_L	d_Q	W_L	d_LW_L	d_Qm_Q		−259.068	−249.567	0.771	0.804
	6	d_L	d_Q	W_L	d_LB	d_LW_L	d_Qm_Q	−264.606	−253.521	0.808	0.841
	6	d_L	m_Q	W_L	d_LW_L	d_Qm_Q	d_QW_L	−264.504	−253.420	0.807	0.840
	1	E_L						−211.787	−208.620	0.276	0.296
	1	d_L						−205.750	−202.583	0.144	0.168
	2	d_L	E_L					−219.611	−214.861	0.432	0.464
S	2	E_L	d_LE_L					−214.218	−209.467	0.340	0.378
L	3	d_L	E_L	d_LE_L				−223.552	−217.218	0.503	0.546
I	3	d_L	E_L	m_QE_L				−223.352	−217.018	0.501	0.543
C	4	d_L	d_Q	E_L	d_Qm_Q			−228.736	−220.818	0.580	0.628
E	4	d_L	m_Q	E_L	d_Qm_Q			−228.429	−220.511	0.577	0.625
	5	d_L	d_Q	E_L	d_Qm_Q	m_QE_L		−235.324	−225.823	0.658	0.707
	5	d_L	m_Q	E_L	d_Qm_Q	m_QE_L		−234.935	−225.434	0.654	0.704
	6	d_L	m_Q	E_L	d_LE_L	d_Qm_Q	m_QE_L	−241.403	−230.318	0.717	0.766
	6	d_L	m_Q	E_L	d_LV	d_LE_L	m_QE_L	−239.618	−228.533	0.703	0.754
	1	d_L						−241.651	−238.484	0.581	0.593
	1	d_Q						−213.883	−210.716	0.093	0.119
	2	d_L	d_Q					−252.120	−247.370	0.694	0.712
G	2	d_L	A_L					−244.534	−239.783	0.623	0.644
R	3	d_L	d_Q	A_L				−257.235	−250.901	0.741	0.764
A	3	d_L	d_Q	C_Q				−255.571	−249.237	0.729	0.752
V	4	d_L	d_Q	A_L	VA_L			−264.066	−256.149	0.791	0.815
E	4	d_L	d_Q	C_Q	A_L			−261.996	−254.079	0.779	0.804
S	5	d_L	d_Q	V	A_L	VA_L		−270.403	−260.902	0.829	0.853
	5	d_L	d_Q	V	C_Q	A_L		−267.812	−258.311	0.816	0.842
	6	d_L	d_Q	V	A_L	d_Qm_L	VA_L	−277.622	−266.537	0.863	0.886
	6	d_L	d_Q	V	C_Q	A_L	C_QP	−275.302	−264.218	0.854	0.879
	1	W_L						−257.017	−253.850	0.373	0.391
	1	W_Q						−243.785	−240.618	0.094	0.120
	2	W_L	W_Q					−262.935	−258.184	0.481	0.511
	2	W_L	NW_L					−260.102	−255.351	0.439	0.471
E	3	W_L	W_Q	NW_L				−267.388	−261.054	0.553	0.591
E	3	W_L	W_Q	J_L				−266.777	−260.443	0.545	0.584
M	4	W_L	J_L	d_LJ_L	d_QJ_L			−278.546	−270.629	0.680	0.716
H	4	W_L	W_Q	J_L	BJ_L			−275.551	−267.633	0.652	0.692
	5	N	W_L	J_L	d_LJ_L	d_QJ_L		−287.171	−277.670	0.754	0.789
	5	W_L	J_L	d_LJ_L	d_QJ_L	m_QJ_L		−284.222	−274.721	0.733	0.771
	6	N	W_L	J_L	d_LJ_L	d_QJ_L	W_LH1L	−292.628	−281.544	0.793	0.828
	6	N	W_L	J_L	d_LJ_L	d_QJ_L	W_LO	−291.852	−280.768	0.788	0.825
	1	E_L						−233.824	−230.657	0.401	0.418
	1	E_Q						−220.094	−216.927	0.123	0.148
	2	E_L	E_Q					−242.387	−237.637	0.540	0.566
E	2	m_L	E_L					−241.624	−236.873	0.530	0.557
E	3	m_L	E_L	E_Q				−254.247	−247.913	0.677	0.705
S	3	m_L	E_L	m_QJ_L				−249.274	−242.940	0.629	0.661
L	4	m_L	E_L	E_Q	d_LE_L			−261.314	−253.397	0.741	0.770
I	4	m_L	E_L	E_Q	m_LK			−258.842	−250.925	0.722	0.754
C	5	m_L	w	E_L	E_Q	d_LE_L		−267.721	−258.220	0.788	0.818
E	5	m_L	E_L	E_Q	d_LE_L	wE_L		−266.078	−256.577	0.778	0.810
	6	m_L	w	E_L	E_Q	d_LE_L	wE_L	−274.498	−263.413	0.828	0.858
	6	m_L	w	E_L	J_Q	m_QJ_L	KJ_Q	−274.386	−263.301	0.828	0.857

Table 3.7: Percent differences of two replicates of MH

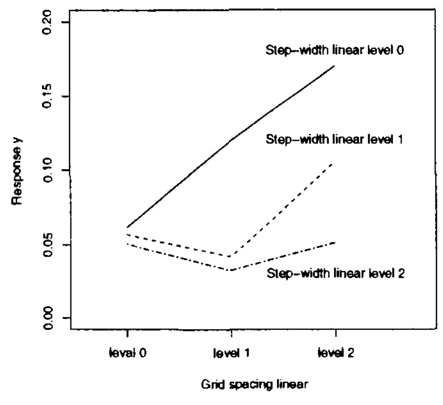
run 1-6	−0.51%	6.75%	−18.94%	−1.40%	−1.45%	−18.00%
run 7-12	−10.82%	−4.62%	−38.77%	−8.34%	−18.77%	182.00%
run 13-18	−4.46%	8.92%	−2.70%	1.05%	6.19%	27.88%
run 19-24	15.08%	−8.09%	−16.23%	−6.33%	2.16%	−0.40%
run 25-30	0.57%	7.29%	6.71%	−52.00%	22.57%	−9.35%
run 31-36	19.15%	−1.73%	−4.61%	7.71%	−39.94%	112.72%

variability of the simulation results across the replicates, we prefer to use the subset chosen from five replicates and try to find the best settings of the important parameters. Note, in a real experimental situation we might have treated the replicates as replicates and used them to estimate the error to determine significance. In our case, we were primarly interested in identifying the most important factors and thus chose this simpler strategy.

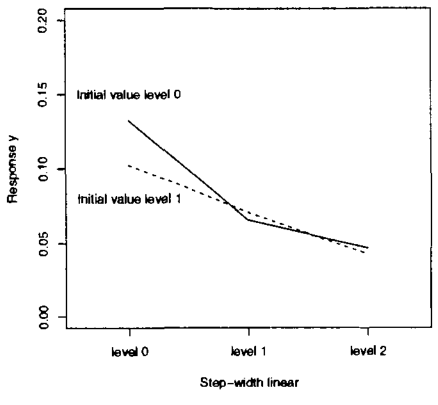
Finding the algorithms' best settings

In this section, we'll try to find the best settings of the important parameters we found in the previous section. The parameters are grouped into noise factors and control factors. In the robust parameter design, noise factors are factors whose values are hard to control during normal process or use conditions, but that can be controlled during the experiment; control factors are factors usually much easier to change. In our experiment, the noise factors are the target distribution parameters d, m and w (denoted by lower-case letters), which in the real world we probably would not know; the control factors are the algorithm parameters, such as number of iterations, N, the step-width of the MH algorithm, etc. (denoted by upper-case letters). In a real situation these are controlled by the analyst. To find the best settings of the control factors, we use techniques for analyzing combined arrays in robust parameter design, which seeks to change the control factor settings to reduce the response variation by exploiting the interaction between control and noise factors (Wu and Hamada (2000)). Control-by-noise interaction plots will be used to choose control factor settings. We also use control main effects and control-by-control interaction plots to find the best combination of the control factors settings which make the response small. Each algorithm will be discussed in turn.

Figure 3.2: MH interaction plots



(a) dW



(b) VW

MH algorithm

Model (3.1) is the MH's linear regression model for the response, $E(y)$ (average AAD), on the significant factors d_L, d_Q, W_L, d_LW_L, d_Lm_Q and VW_L,

$$y = 0.077 + 0.037d_L + 0.0147d_Q - 0.051W_L - 0.046d_LW_L + 0.031d_Lm_Q + 0.018VW_L. \quad (3.1)$$

Among these effects, d_L, d_Q, m_Q are the noise factors, W_L, V are the control effects. d_LW_L is a noise-control factor interaction. The interaction plots for dW and VW are shown in Figure 3.2. Figure 3.2(a) recommends level 2 of W since it has the smallest response variation over setting of d and the smallest average y response (main effect). Figure 3.2(b) suggests level 1 for V is the better setting, since the combination of V at level 1 and W at level 2 yields the smallest y response. The other control factors do not appear to impact the MH algorithm.

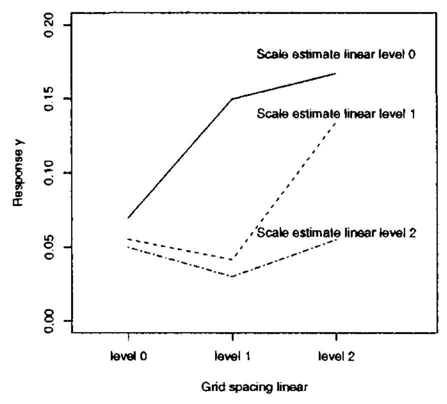
Slice sampling

Model (3.2) is Slice sampling's linear regression model on the factors d_L d_Q, E_L, d_LE_L, d_QE_L, d_QE_Q,

$$y = 0.084 + 0.043d_L + 0.012d_Q - 0.059E_L - 0.046d_LE_L + 0.031d_QE_L - 0.038d_QE_Q. \quad (3.2)$$

The noise effects are d_L, d_Q and the control effects are E_L and E_Q. The control-by-noise interaction plot of dE is given in Figure 3.3. The plot suggests level 2 is the best choice of

Figure 3.3: Slice sampling interaction plots



(a) dE

E since it has the smallest response average (main effect) and also the smallest variation across settings of d.

Graves' Method

Model (3.3) is the Graves' method linear regression model on the factors d_L, d_Q, A_L, $d_L m_L$, $d_L V$ and $d_L A_L$,

$$y = 0.088 + 0.0702 d_L + 0.028 d_Q + 0.019 A_L + 0.049 d_L m_L - 0.037 d_L V + 0.035 d_L A_L. \quad (3.3)$$

d_L, d_Q and m_L are the noise factors. A_L and V are control factors. Figure 3.4(a) is the interaction plot of dA. It suggests level 0 is the best choice of A since it has the smallest response average and variation over settings of d. Figure 3.4(b) suggests level 1 is the better choice of V for similar reasons.

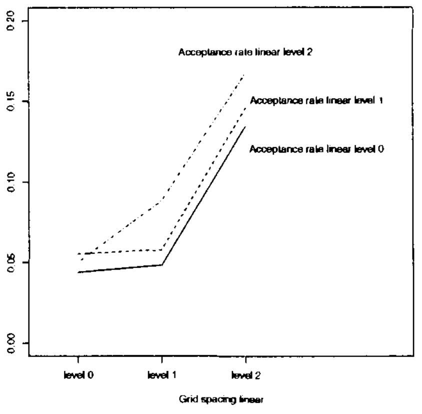
EE-MH

Model (3.4) is the EE-MH linear regression model on factors W_L, W_Q, $d_L W_L$, $d_L W_Q$, $m_L W_L$, $W_L O$,

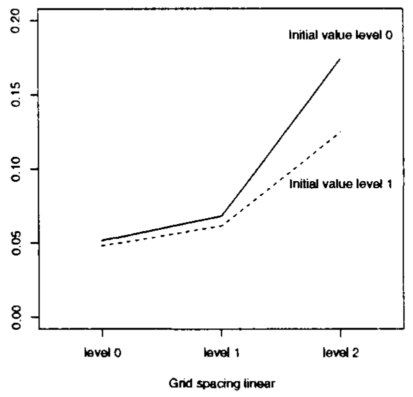
$$y = 0.066 - 0.045 W_L + 0.027 W_Q - 0.028 d_L W_L + 0.029 d_L W_Q - 0.027 m_L W_L + 0.015 W_L O. \quad (3.4)$$

d_L and m_L are the noise effects. W_L, W_Q and O are the control effects. Figure 3.5(a) gives the interaction plot for dW while Figure 3.5(b) the interaction plot for mW. Both suggest W at level 0 is worse than levels 1 or 2, but it's hard to tell which one is better between

Figure 3.4: Graves' method interaction plots



(a) dA



(b) dV

levels 1 and 2. Figure 3.5(c) shows that if W is set at level 1, O should be set at level 1, while if W is set at level 2, O should be set at level 0. Either choice looks fine. We choose W at level 1 and O at level 1.

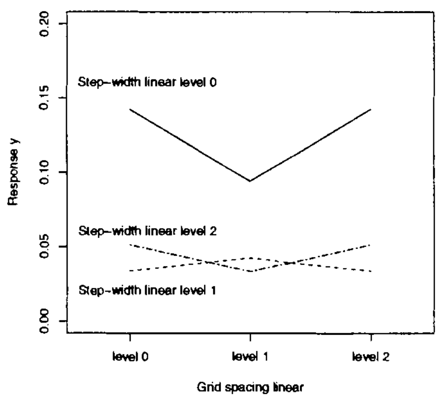
EE-Slice

Model (3.5) is EE-Slice linear regression model on factors m_L, N, E_L, E_Q, m_L and NE_L,

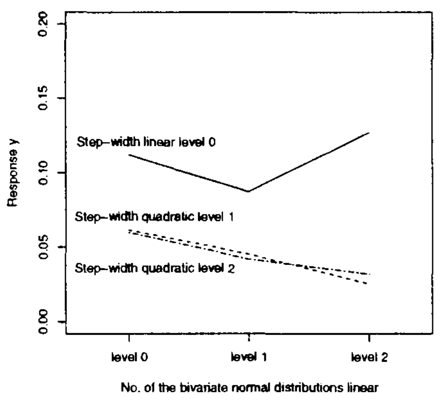
$$y = 0.072 - 0.015m_L - 0.009N - 0.061E_L + 0.036E_Q - 0.017m_Lw + 0.021NE_L. \quad (3.5)$$

m_L is the noise factor, E_L, E_Q and N are the control factors. Viewing Figure 3.6, N at level 1 and E at level 1 performs best.

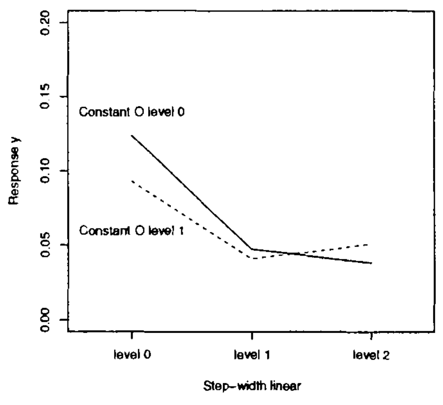
Figure 3.5: EE-MH interaction plots



(a) dW



(b) mW

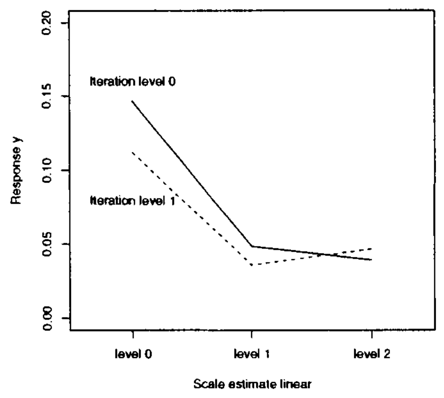


(c) WO

Table 3.8: Top 2 weak hereditary subsets of the algorithm on average response

	Size	Selected factors						*AIC*	*BIC*	*Adjr2*	*Rsq*
	1	W_L						−236.079	−232.912	0.390	0.408
	1	d_L						−225.880	−222.713	0.191	0.214
	2	d_L	W_L					−250.185	−245.435	0.598	0.621
	2	W_L	d_LW_L					−244.403	−239.653	0.528	0.555
	3	d_L	W_L	d_LW_L				−265.981	−259.647	0.747	0.769
M	3	d_L	W_L	d_LB				−252.566	−246.232	0.633	0.665
H	4	d_L	d_Q	W_L	d_LW_L			−269.670	−261.753	0.777	0.803
	4	d_L	W_L	d_LV	d_LW_L			−267.911	−259.993	0.766	0.793
	5	d_L	d_Q	W_L	d_LW_L	d_Qm_Q		−274.237	−264.736	0.808	0.836
	5	d_L	d_Q	W_L	d_LV	d_QW_L		−272.318	−262.817	0.798	0.827
	6	d_L	d_Q	W_L	d_LW_L	d_Lm_Q	VW_L	−277.961	−266.877	0.831	0.860
	6	d_L	d_Q	W_L	d_LV	d_LW_L	d_QW_Q	−275.530	−264.446	0.819	0.850
	1	E_L						−227.084	−223.917	0.399	0.416
	1	d_L						−216.544	−213.377	0.195	0.218
	2	d_L	E_L					−241.902	−237.152	0.612	0.634
	2	E_L	d_LE_L					−230.725	−225.975	0.471	0.501
S	3	d_L	E_L	d_LE_L				−249.378	−243.044	0.692	0.719
L	3	d_L	B	E_L				−244.029	−237.695	0.643	0.674
I	4	d_L	E_L	d_LE_L	d_QE_L			−252.621	−244.703	0.726	0.757
C	4	d_L	E_L	d_LV	d_LE_L			−251.315	−243.397	0.715	0.748
E	5	d_L	d_Q	E_L	d_LE_L	d_QE_Q		−256.641	−247.140	0.760	0.794
	5	d_L	d_Q	E_L	d_LE_L	d_Qm_Q		−255.549	−246.048	0.753	0.788
	6	d_L	d_Q	E_L	d_LE_L	d_QE_L	d_QE_Q	−262.019	−250.934	0.798	0.832
	6	d_L	d_Q	E_L	d_LE_L	d_Qm_Q	d_QE_L	−260.683	−249.598	0.790	0.826
	1	d_L						−239.892	−236.725	0.578	0.590
	1	d_Q						−211.372	−208.205	0.069	0.095
	2	d_L	d_Q					−247.425	−242.674	0.667	0.686
G	2	d_L	A_L					−241.814	−237.063	0.610	0.633
R	3	d_L	d_Q	A_L				−250.625	−244.291	0.702	0.728
A	3	d_L	d_Q	d_LC_L				−250.095	−243.761	0.698	0.724
V	4	d_L	d_Q	d_Lm_L	d_LV			−254.980	−247.062	0.743	0.772
E	4	d_L	d_Q	A_L	d_LC_L			−254.079	−246.161	0.736	0.766
S	5	d_L	C_Q	A_L	d_LV	m_LA_L		−261.326	−251.825	0.789	0.819
	5	d_L	d_Q	A_L	d_Lm_L	d_LV		−260.362	−250.861	0.783	0.814
	6	d_L	d_Q	A_L	d_Lm_L	d_LV	d_LA_L	−267.563	−256.478	0.826	0.856
	6	d_L	d_Q	C_Q	A_L	d_LV	m_LA_L	−267.342	−256.257	0.825	0.855
	1	W_L						−255.811	−252.644	0.469	0.484
	1	W_Q						−238.576	−235.409	0.142	0.167
	2	W_L	W_Q					−267.852	−263.101	0.629	0.650
	2	W_L	W_LO					−257.878	−253.128	0.511	0.539
E	3	W_L	W_Q	d_LW_Q				−273.168	−266.834	0.688	0.715
E	3	W_L	W_Q	W_LO				−272.034	−265.700	0.678	0.706
M	4	W_L	W_Q	d_LW_Q	W_LO			−278.903	−270.986	0.740	0.770
H	4	W_L	W_Q	d_LW_L	m_LW_L			−278.078	−270.160	0.734	0.765
	5	W_L	W_Q	d_LW_L	d_LW_Q	m_LW_L		−287.557	−278.056	0.800	0.829
	5	W_L	W_Q	d_LW_Q	NW_L	W_LO		−285.015	−275.514	0.786	0.816
	6	W_L	W_Q	d_LW_L	d_LW_Q	m_LW_L	W_LO	−290.968	−279.883	0.822	0.853
	6	B	W_L	W_Q	d_LW_Q	d_QB	BJ_L	−290.159	−279.074	0.818	0.849
	1	E_L						−246.597	−243.430	0.555	0.568
	1	E_Q						−224.214	−221.047	0.172	0.196
	2	E_L	E_Q					−266.344	−261.593	0.750	0.764
E	2	m_L	E_L					−247.603	−242.852	0.579	0.603
E	3	m_L	E_L	E_Q				−270.048	−263.714	0.780	0.799
S	3	E_L	E_Q	NE_L				−269.837	−263.503	0.778	0.797
L	4	m_L	E_L	E_Q NE_L				−274.576	−266.658	0.810	0.832
I	4	E_L	E_Q	d_LE_Q	NE_L			−273.658	−265.741	0.805	0.828
C	5	m_L	E_L	E_Q	m_Lw	NE_L		−277.426	−267.925	0.829	0.853
E	5	m_L	N	E_L	E_Q	NE_L		−277.115	−267.614	0.827	0.852
	6	m_L	N	E_L	E_Q	m_Lw	NE_L	−280.670	−269.586	0.847	0.873
	6	m_L	E_L	E_Q	H_{1Q}	d_QH_{1Q}	m_Lw	−280.536	−269.451	0.846	0.873

Figure 3.6: EE-MH control-by-noise interaction plots



Chapter 4

Simulation Study

4.1 Compare the Algorithms at Their Best Settings

In this section, a small simulation will be done to compare the algorithms at their best settings. Only the target distribution parameters will be varied in this simulation experiment. From the screening experiment, we found the distribution parameters d and m appear with high frequency in the 12 best weak hereditary models of all the five algorithms, while the other parameter related to the target distributions, w, does not. So we set the distribution to have equal weights, and only use parameters d and m with their previous levels to control the distribution shape. For the algorithm parameters, the settings are fixed at the best values chosen using the robust parameter design concepts of the previous chapter. The simulation in this section is based on a 9-run 3^2 full factorial design. This design matrix is given with the simulation results in Appendix B. For each algorithm, the simulation is repeated to get five replicates.

The algorithm parameter settings are as follows. The common algorithm parameters, iteration number N and burn-in length B are both set at their previous high level: 40000 and 2000 iterations, respectively. These high settings should give the algorithms a better chance to converge. The initial value V is set to 0, since MH and Graves method both perform better at this setting. For the algorithm specific parameters, the important ones from the screening experiment are set as follows: the step-width W of MH is set to level 2, 5.1. The scale estimate E of Slice sampling is set to level 2, 5.1. The Acceptance rate A of Graves method is set to level 0, 0.2. The step-width W of EE-MH is set to level 1, 2.6. The constant O of EE-MH is set to level 1, 4. The scale estimate E of EE-Slice is set to level 1, 2.6. The iteration number N is set 40000 which is consistent with the common parameter setting. The other algorithm specific parameter settings are: number of the step-widths P is set as 11, which compared to 5 we think might give better chance to fit the model (2.1) accurately. The initial guess of the optimal step-width is set to 2, using the previous middle level. For the two EE algorithms, the number of energy levels, K, is set to 2, the EE-jump probability J is set to 0.2, energy level 1 is set to 3, and Constant O is set to 4. The summary of these control parameter settings is given in Table 4.1.

Table 4.1: Best parameter settings, used in the comparison simulation

	Common algorithm parameters		
	Iteration number	N	40000
	Iteration number in burn-in	B	2000
	Initial value	V	0
	Algorithm specific parameters		
MH	Step-width	W	5.1
Slice	Scale parameter	E	5.1
Graves	Acceptance rate	A	0.2
	Number of the step-widths	P	11
	Initial guess of the optimal step-width	C	2
EE-MH	Step-width	W	2.6
	Constant	O	4
	Probability of EE jump	J	0.3
	Energy level 1	H_1	3
	Number of energy levels	K	2
EE-Slice	Scale parameter	E	2.6
	Constant	O	4
	Probability of EE jump	J	0.3
	Energy level 1	H_1	3
	Number of energy levels	K	2

4.2 9-Run Simulation Results

The complete results of the 9-run simulation are shown in Tables B.1 to B.5, in Appendix B. Here we list the mean and the standard deviation of the response for each run (Table 4.2). We can see, for the nine combinations of d and m, the performance of Graves' method is not very stable. In runs 1,2,4,5,7 and 8 it performs well, but in runs 3,6 and 9 it performs badly.

Graves' method performs badly when d is large, that is, when the modes are separated and far apart. Basically, if the MH gets trapped in a single mode during burn-in, Graves' method will only give a good step-width for exploring the single mode and not the entire space. When the modes are overlapping or connected, Graves' method works well.

Table 4.2: 9 run simulation results of MH

	Parameters		MH		Slice		Graves		EE-MH		EE-Slice	
Run	d	m	$\bar{y}_{mh}$	sd_{mh}	$\bar{y}_{sl}$	sd_{sl}	$\bar{y}_{gr}$	sd_{gr}	$\bar{y}_{eemh}$	sd_{eemh}	$\bar{y}_{eesl}$	sd_{eesl}
1	1	3	0.0601	0.0048	0.0588	0.0006	0.0593	0.0019	0.0598	0.0014	0.0579	0.0008
2	3	3	0.0553	0.0020	0.0560	0.0009	0.0545	0.0022	0.0559	0.0016	0.0558	0.0012
3	5	3	0.0553	0.0051	0.0553	0.0012	0.1067	0.0324	0.0561	0.0023	0.0553	0.0012
4	1	5	0.0469	0.0011	0.0450	0.0004	0.0450	0.0011	0.0468	0.0014	0.0444	0.0008
5	3	5	0.0351	0.0007	0.0349	0.0007	0.0357	0.0009	0.0371	0.0011	0.0342	0.0007
6	5	5	0.0368	0.0022	0.0365	0.0016	0.1585	0.0494	0.0366	0.0018	0.0358	0.0008
7	1	7	0.0361	0.0018	0.0348	0.0005	0.0352	0.0022	0.0366	0.0004	0.0349	0.0007
8	3	7	0.0215	0.0009	0.0207	0.0002	0.0219	0.0011	0.0216	0.0007	0.0209	0.0003
9	5	7	0.0298	0.0095	0.0391	0.0180	0.1372	0.0298	0.0221	0.0005	0.0241	0.0031

For the other four algorithms, among these nine runs, they all work reasonably well. The average absolute distance range between the CDF and ECDF is $(0.02, 0.06)$. The standard deviation shows that the two EE algorithms have better stability than MH and Slice sampling.

If we group the 9 runs into three groups, 1-3, 4-6 and 7-9, and look at the results of the four algorithms closely, we can see every first run of the three groups have worse performance than other two runs. Those cases are when $d = 1$, which shows the choice of the step-width or the scale parameter, 5.1, is somewhat too large compared to the grid spacing $d = 1$. As the grid spacing d and the number of normal distributions m increase, the performance of the four algorithms are good based on the mean and the standard deviation of the five replicates' response, until run 9 ($d = 5$ and $m = 7$). For this run, MH and Slice sampling algorithms have worse repeatability (sd) compared to other settings, and compared to the EE algorithms. This is an interesting phenomenon. It suggests that when there are more modes far apart, and the step-width or scale parameter is big enough to cut across the modes, this creates more variability in the simulation results. This motivates us to explore cases where d gets even bigger. We would expect for MH and slice sampling, the performance level would decrease because of the sensitivity to the step-width or the scale parameter. For EE algorithms, the performance will also be affected by the step-width or the scale parameter, but the effect might be more mild than MH or slice sampling, since the EE algorithm starts from the flatter distributions and gradually moves on to the target distribution.

Another important factor to be considered in the application of the algorithms in the real world is the simulation time. The recorded run times of the five algorithms is given in Table B.6 in Appendix B. From Table B.6, we can see that for the MH and EE algorithms, the simulation time of run 1-3 is similar, run 4-6 is similar, and run 7-9 is similar. This is because m is set at three different levels 3, 5, and 7. Since the target distribution $f(\boldsymbol{x})$ is the sum of bivariate normal distributions, as m increases, the computation time of $f(\boldsymbol{x})$ increases. Every generated point is related to the computation of $f(\boldsymbol{x})$. For example, in the MH algorithm whether a new generated point $\boldsymbol{x}^*$ is accepted depends on whether a random number from $Unif(0,1)$ is less than $\alpha(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*) = \min\left[(f(\boldsymbol{x}^*)q(\boldsymbol{x}^*, \boldsymbol{x}^{(i)}))/(f(\boldsymbol{x}^{(i)})q(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*)), 1\right]$.

With the number of iterations $N = 40000$ and the number of iterations in burn-in period $B = 2000$, the MH algorithm performs fastest, which taking less than 30 seconds. Graves' method is just slightly slower. Slice sampling takes less than 76 seconds. While the EE algorithms take about 20 minutes. The EE-MH algorithm takes about 45 times as much time as that of the MH algorithm, while the EE-Slice algorithm takes about 25 times that of the Slice sampling algorithm. This raises an interesting issue: From previous discussion, we might expect the EE algorithms to be less sensitive to the step-width or choice of scale parameter than the MH or slice sampling algorithms. But, the EE algorithms take much longer to run. If given the same length of time as the EE algorithm, what will the simulation performance of the MH and slice sampling algorithms be like? Would a long simulation time overcome the problem of the sensitivity to the step-width or scale parameter?

To summarize the above, there is little difference among most of the algorithms based on the 9-run simulation. But two further problems have been identified:

1. The sensitivity to the step-width or the scale parameter of the algorithms when d keeps increasing.

2. The simulation performance of MH and slice sampling compared with EE algorithms if given the same simulation time.

4.3 Simulations for Fixed Time

To explore these two problems, we add two more runs after the previous 9-run settings: one run with $d = 7$ and $m = 7$, and one run with $d = 9$ and $m = 7$. The resulting 11-run experiment was carried out for all five methods, aiming to run all algorithms for about the same length of time: 20 minutes. For the MH algorithm, Slice sampling, and Graves' method, run time was controlled to 20 minutes by stopping after an appropriate number of iterations. Because of the special structure of the EE algorithms, running several parallel chains continuously, it was not possible to control run time by repeating shorter sets of iterations. So we let the EE algorithms run for $N = 40000$ iterations, which takes from 11 to 21 minutes for EE-MH and from 16 to 26 minutes for the EE-Slice algorithm.

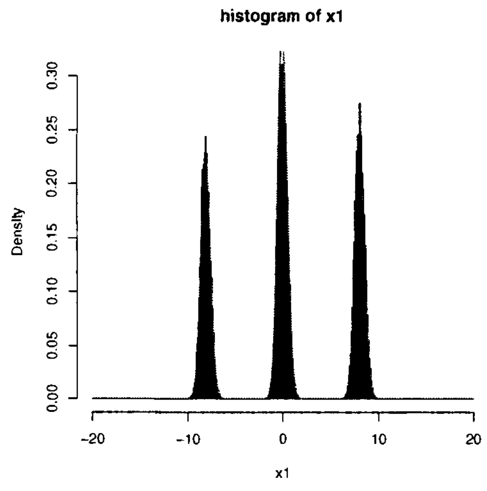
The results of one replicate of the 11 runs are given in Table 4.3. The number of iterations of MH, slice sampling and Graves' method are also listed in Table 4.3. MH and Graves' method both had over 3 million , 2 million and 1 million iterations, for cases with $m = 3$, $m = 5$ and $m = 7$, respectively. The iteration range of Slice sampling is $(0.6, 1.4)$ million. For the overall performance of the five algorithms across the first 9 runs, we see from the mean and the standard deviation of the response that the EE algorithms no longer show superior performance to the MH algorithm or Sliced sampling. This is also true in comparison to Graves' method except for run 9.

If we look closely at the results of each run, it is easy to see for the first 8 runs, the five algorithms perform similarly well. For run 9, which is $d = 5$ and $m = 7$, Graves' method seems to perform worse than other algorithms. Recall that these 9 settings were used to set the tuning parameters of the algorithms.

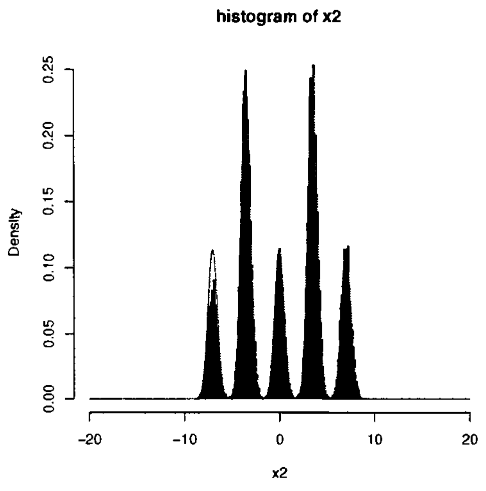
When d keeps increasing to 7, which is run 10, Slice sampling and Graves' method don't perform well. The MH algorithm and the two EE algorithms seem to work well. When d increases to 9, the MH algorithm's performance also deteriorates, while the two EE algorithms still perform reasonably well.

Histogram plots of each variate give more straightforward comparison of the performance of the five algorithms. Since the five algorithms perform similarly well for the first 9 runs (except run 9 of Graves' method), we will skip showing the histogram plots of those runs. For run 10, MH, EE-MH and EE-Slice perform well, while slice sampling and Graves' method seem to perform worse. Here we will display the histogram plots for various runs.

Figure 4.1: Histogram plots of two variables of MH for run 10

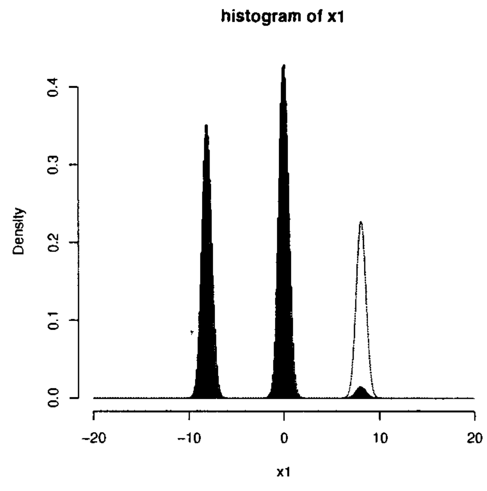


(a) x_1

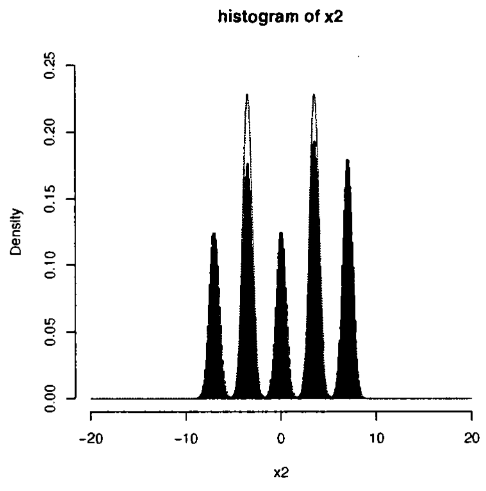


(b) x_2

Figure 4.2: Histogram plots of two variables of slice sampling for run 10



(a) x_1



(b) x_2

Table 4.3: 11 – *run* simulation results

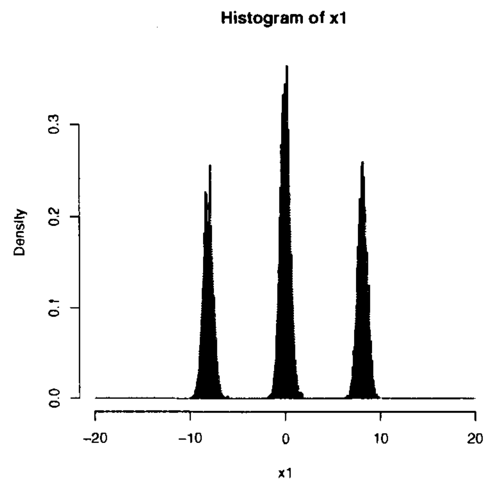
	Parameters		Iteration (million)			Response				
Run	d	m	N_{mh}	N_{sl}	N_{gr}	y_{mh}	y_{sl}	y_{gr}	y_{eemh}	y_{eesl}
1	1	3	3.48	1.37	3.6	0.0587	0.0586	0.0583	0.0600	0.0582
2	3	3	3.58	1.30	3.6	0.0558	0.0556	0.0540	0.0530	0.0559
3	5	3	3.58	1.23	3.6	0.0556	0.0565	0.0537	0.0556	0.0573
4	1	5	2.34	1.05	2.36	0.0439	0.0451	0.0438	0.0479	0.0456
5	3	5	2.34	0.94	2.36	0.0334	0.0338	0.0341	0.0368	0.0358
6	5	5	2.34	0.86	2.36	0.0353	0.0347	0.0373	0.0371	0.0357
7	1	7	1.74	0.87	1.76	0.0361	0.0325	0.0350	0.0373	0.0344
8	3	7	1.74	0.74	1.76	0.0201	0.0206	0.0204	0.0222	0.0201
9	5	7	1.74	0.67	1.76	0.0219	0.0241	0.0711	0.0223	0.0220
					$\bar{y}_{1-9}$	0.0401	0.0402	0.0453	0.0414	0.0406
					$sd_{y,1-9}$	0.0144	0.0143	0.0154	0.0138	0.0146
10	7	7	1.74	0.66	1.76	0.0261	0.0764	0.1534	0.0237	0.0240
11	9	7	1.74	0.66	1.76	0.1432	0.1509	0.1516	0.0253	0.0251
					$\bar{y}_{10-11}$	0.0846	0.1136	0.1525	0.0245	0.0246
					$sd_{y,10-11}$	0.0828	0.0527	0.0013	0.0012	0.0007

Figures 4.1 to 4.3 give histograms of the two variables for samples from the MH algorithm, slice sampling and the EE-MH sampling for run 10, respectively (EE-Slice performed similar to EE-MH). The solid lines are the marginal distribution plots. For MH and EE-MH, the histogram plots fit the marginal distribution plot fairly well, while slice sampling, Figure 4.2(a) shows one mode of x_1 is not sampled as it should be, while the other two are over sampled. Similar things happen to x_2, as well.

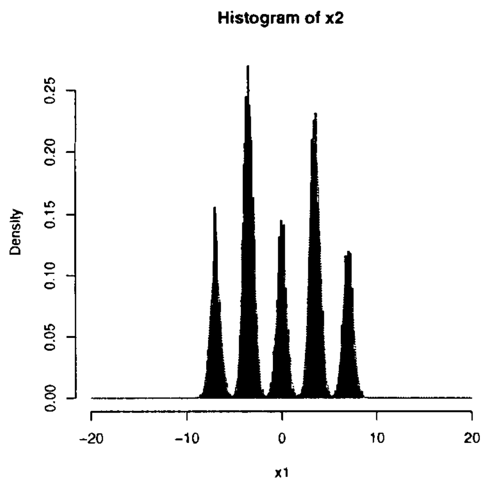
When d increases to 9, which is the case in run 11, the histogram plots of x_2 for MH, slice sampling and Graves' method and EE-Slice are given in Figure 4.4, while Figure 4.5 gives the three parallel chains using EE-MH. We can see that the EE algorithms continue to perform quite well, while the others have a tendency to get stuck in a single mode. This is because the tuning parameters that control step-width are too small.

The above exploration shows, Graves' method seems to work fine in what amount to nearly uni-modal situations, but when there are multiple, well-separated modes, it does not do well as a method of recommending a step-width for the MH algorithm. The MH algorithm and slice sampling are more sensitive to the step-width or scale parameter than the EE algorithms, but the EE algorithms take much longer to run. Given a longer simulation time, MH and slice sampling improve greatly. Overall, the EE algorithms seem to have more stable performance than the other algorithms.

Figure 4.3: Histogram plots of two variables of EE-MH for run 10

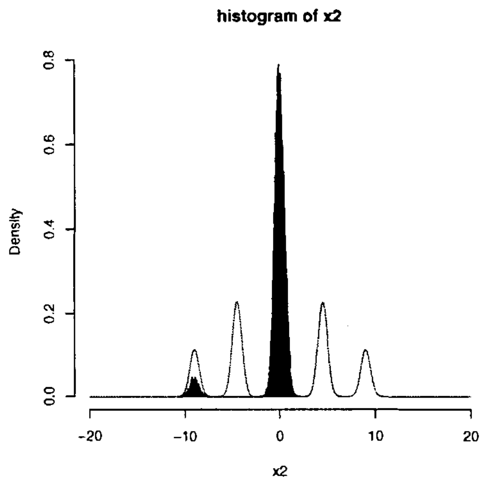


(a) x_1

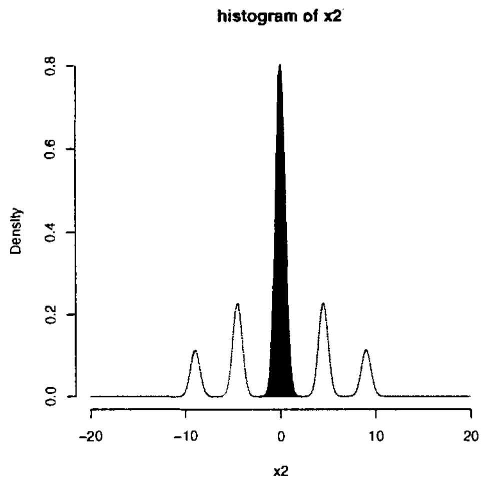


(b) x_2

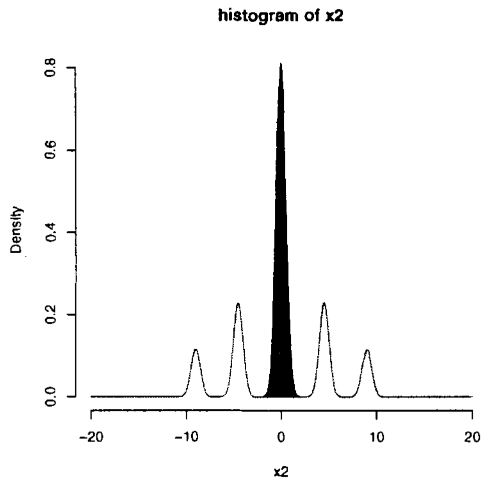
Figure 4.4: Histogram plots of x_2 of four algorithms for run 10



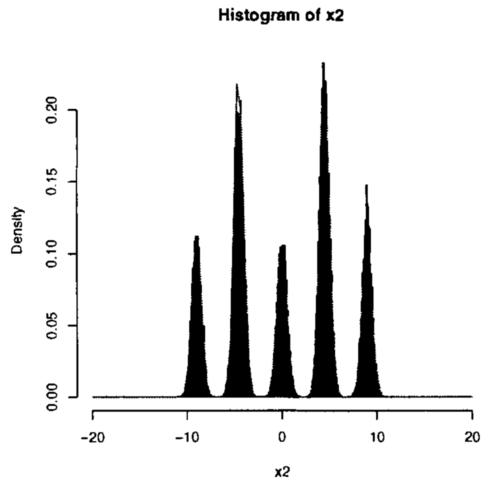
(a) MH x_2



(b) Slice sampling x_2

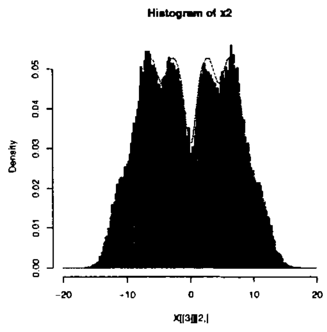


(c) Graves' method x_2

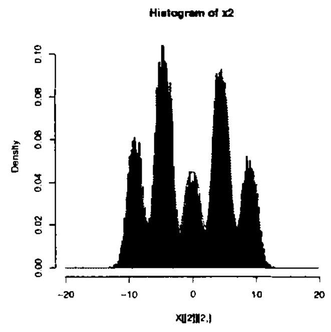


(d) EE-Slice x_2

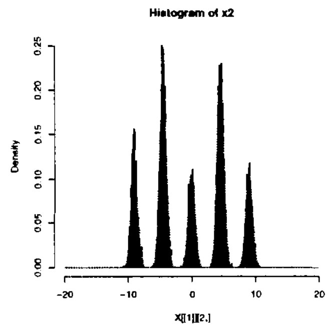
Figure 4.5: Histogram plot of x_2 of EE-MH three chains for run 11



(a) Chain $x_1^{(2)}$



(b) Chain $x_1^{(1)}$



(c) Chain $x_1^{(0)}$

Chapter 5

Conclusions

5.1 Summary

The sampling problem we are trying to solve is to sample a multi-dimensional distribution that has several modes. The most well-known sampling algorithm is the MH algorithm, but it has some difficulty with this type of problem because of its sensitivity to the tuning parameter, step-width. When the step-width is too small, it has difficulty cutting across the modes. Three alternative algorithms recently introduced in the literature, Graves' method, slice sampling and EE sampling, are expected to solve part of the problem. The summary of these algorithms are listed as follows.

Graves' method

Graves' method is basically an improved MH algorithm. The difference between these two algorithms is, Graves' method attempts to automatically find a better choice of the step-width in the burn-in period and applies it in the MH algorithm. It does so by using the linear relationship of the logit of the acceptance rate (A) and the log of the step-width (W), a step-width is suggested by an ideal acceptance rate through this model. Graves' method is shown to improve the MH algorithm in some situation, for example, when the target distribution has a single mode or several connected modes. But when the modes are far apart, which is a difficult situation for the MH algorithm in general, Graves' method does not show improvement over the MH algorithm. There are two points which might be related to this problem: (i) The assumption of Graves' method that finding a desirable step-width can be solved by some connection with achieving a desirable acceptance rate; and (ii) The accuracy of the linear relationship of the logit of the acceptance rate (A) and the log of the step-width (W) in more extreme situations.

Slice sampling

Slice sampling is another algorithm that uses tuning parameters, scale parameter, which plays a similar role to the step-width in the MH algorithm. Our exploration suggests that the performance of slice sampling is sensitive to this scale parameter. When there are more modes, and the scale parameter is too small to slice across the modes, it fails to perform well. One point that needs to be noted is that the slicing strategy applied in this project is not the only option available. Since the target distribution is a bivariate distribution, the slicing strategy we used is locating the hyper-rectangle which includes the current point by setting a common random scale parameter E on each dimension. So the slice interval is restricted to the length of E. It is not hard to see that when E is too small this strategy will face the problem of not cutting across the modes. There are more slicing strategies introduced in Neal (2003). For example, the "stepping out" procedure, which creates the chance to extend the slice interval from the length of E.

EE algorithms

The EE algorithm generates a new point by an MH local move or by jumping to a point that has the similar energy level (value of $f(\boldsymbol{x})$) as the current point. It starts from a chain aiming at a flatter distribution which corresponds to a higher energy level, and moves on to the target distribution corresponding to a lower energy level. The EE-jump and parallel chains are designed to help overcome the local traps (modes) of the distribution. In this project, two EE algorithms, one with the usual MH local move and one with the slice sampling local move, were explored. The EE algorithms were shown to have more stable performance compared to the other three algorithms. When the distribution has separated modes, the EE algorithms are less sensitive to the tuning parameters than the MH and slice sampling algorithms. But the disadvantage of EE algorithms are in taking a much longer time to run. This is an important problem in practice. One point needs needs to also be noted is that for the parameter settings of EE algorithm. one has to construct the energy levels and temperature levels, which is related to some prior knowledge of the distribution. This creates some complexity in applying this algorithm.

As a summary of these algorithms, if not considering the time problem, the EE algorithms would be a stable choice to sample a distribution. Especially if it is known that the distribution has several modes, but one is not sure how they are positioned. However from an application standpoint, since MH and slice sampling perform much faster, one might argue that some preliminary exploration could reveal a good choice for the tuning parameters followed by a much larger number of iterations, might yield similar performance. It seems the big problem happens when there are separate modes and the tuning parameter is too small. So people may spend the time to try bigger tuning parameter to explore what the right values are, and then apply them. And how this can be done might refer to each specific problem.

5.2 Future Work

There are some points that might be interesting to improve or try out in future.

1. For slice sampling, as we mentioned above, the slicing strategy in this project is locating the hyper-rectangle which includes the current point by setting a common random scale parameter E on each dimension. This limits the slice interval to the length of E. More slicing strategies would be interesting to try. For example, the "stepping out" procedure, which creates the chance to extend the slice interval from the length of E. Although it might be time consuming to apply it in higher dimensions.

2. For the EE algorithms, we used the suggestions from Kou, Zhou, and Wong (2006) for the parameter settings. But we are not sure if there is a better way to set these parameters, or how to find a better way to set these parameters.

Appendix A

Screening Experiment Result

Table A.1: 36 run design matrix and simulation results of MH

	Design matrix							Simulation results						
Run	d	m	w	V	N	B	W	y_{mh1}	y_{mh2}	y_{mh3}	y_{mh4}	y_{mh5}	$\bar{y}_{mh}$	sd_{mh}
1	1	1	0	0	0	0	2	0.0382	0.0381	0.0406	0.0378	0.0367	0.0383	0.0014
2	1	1	1	1	0	0	2	0.0401	0.0428	0.0418	0.0491	0.0478	0.0443	0.0039
3	2	2	0	0	1	1	1	0.1174	0.0952	0.1902	0.1537	0.0630	0.1239	0.0497
4	2	2	1	1	1	1	1	0.1025	0.1011	0.1379	0.0986	0.1880	0.1256	0.0385
5	1	2	0	0	1	0	1	0.0226	0.0222	0.0221	0.0198	0.0204	0.0214	0.0012
6	1	2	1	0	0	1	1	0.0320	0.0262	0.0377	0.0246	0.0253	0.0292	0.0056
7	2	1	0	1	0	0	2	0.0422	0.0377	0.0516	0.0443	0.0360	0.0424	0.0061
8	2	1	1	1	1	0	2	0.0444	0.0424	0.0416	0.0452	0.0446	0.0436	0.0015
9	1	2	0	1	0	1	2	0.0319	0.0195	0.0239	0.0271	0.0223	0.0249	0.0048
10	1	2	1	0	1	0	2	0.0228	0.0209	0.0247	0.0228	0.0243	0.0231	0.0015
11	2	1	0	1	1	1	1	0.0658	0.0534	0.0405	0.0597	0.1213	0.0681	0.0312
12	2	1	1	0	0	1	1	0.0638	0.1799	0.1086	0.0658	0.1039	0.1044	0.0471
13	2	2	0	0	0	0	0	0.2110	0.2016	0.1864	0.2111	0.2250	0.2070	0.0142
14	2	2	1	1	0	0	0	0.1360	0.1481	0.1520	0.1520	0.1530	0.1482	0.0071
15	0	0	0	0	1	1	2	0.0579	0.0564	0.0613	0.0613	0.0582	0.0590	0.0022
16	0	0	1	1	1	1	2	0.0660	0.0667	0.0657	0.0635	0.0719	0.0668	0.0031
17	2	0	0	0	1	0	2	0.0573	0.0608	0.0586	0.0575	0.0590	0.0586	0.0014
18	2	0	1	0	0	1	2	0.0560	0.0716	0.0568	0.0580	0.0670	0.0619	0.0070
19	0	2	0	1	0	0	0	0.0638	0.0735	0.0543	0.0754	0.1115	0.0757	0.0217
20	0	2	1	1	1	0	0	0.0479	0.0440	0.0523	0.0642	0.0465	0.0510	0.0080
21	2	0	0	1	0	1	0	0.1348	0.1130	0.1226	0.1266	0.1188	0.1232	0.0082
22	2	0	1	0	1	0	0	0.2163	0.2026	0.2056	0.1874	0.1976	0.2019	0.0106
23	0	2	0	1	1	1	2	0.0356	0.0364	0.0351	0.0351	0.0326	0.0350	0.0014
24	0	2	1	0	0	1	2	0.0426	0.0424	0.0411	0.0416	0.0392	0.0414	0.0014
25	0	0	0	0	0	0	1	0.0618	0.0622	0.0591	0.0620	0.0647	0.0620	0.0020
26	0	0	1	1	0	0	1	0.0666	0.0714	0.0664	0.0665	0.0706	0.0683	0.0025
27	1	1	0	0	1	1	0	0.1887	0.2013	0.1647	0.0797	0.0519	0.1373	0.0673
28	1	1	1	1	1	1	0	0.1766	0.0848	0.0774	0.0872	0.0665	0.0985	0.0444
29	0	1	0	0	1	0	0	0.0385	0.0472	0.0592	0.0578	0.0385	0.0482	0.0101
30	0	1	1	0	0	1	0	0.0790	0.0716	0.0619	0.1035	0.0541	0.0740	0.0190
31	1	0	0	1	0	0	1	0.0567	0.0675	0.0588	0.0564	0.0521	0.0583	0.0057
32	1	0	1	1	1	0	1	0.0593	0.0582	0.0586	0.0613	0.0593	0.0593	0.0012
33	0	1	0	1	0	1	1	0.0437	0.0417	0.0470	0.0444	0.0482	0.0450	0.0026
34	0	1	1	0	1	0	1	0.0515	0.0555	0.0543	0.0524	0.0536	0.0535	0.0016
35	1	0	0	1	1	1	0	0.1422	0.0854	0.1532	0.1281	0.0785	0.1175	0.0337
36	1	0	1	0	0	1	0	0.0958	0.2038	0.1158	0.1065	0.1065	0.1257	0.0442

Table A.2: 36 run design matrix and simulation results of slice sampling

	Design matrix							Simulation results						
Run	d	m	w	V	N	B	E	y_{sl1}	y_{sl2}	y_{sl3}	y_{sl4}	y_{sl5}	$\bar{y}_{sl}$	sd_{sl}
1	1	1	0	0	0	0	2	0.0338	0.0356	0.0328	0.0367	0.0361	0.0350	0.0016
2	1	1	1	1	0	0	2	0.0391	0.0450	0.0431	0.0413	0.0442	0.0425	0.0024
3	2	2	0	0	1	1	1	0.1837	0.1874	0.2108	0.1808	0.1066	0.1738	0.0394
4	2	2	1	1	1	1	1	0.1069	0.1153	0.1022	0.0891	0.1160	0.1059	0.0110
5	1	2	0	0	1	0	1	0.0252	0.0220	0.0204	0.0255	0.0238	0.0234	0.0022
6	1	2	1	0	0	1	1	0.0231	0.0321	0.0226	0.0266	0.0365	0.0282	0.0060
7	2	1	0	1	0	0	2	0.0368	0.0928	0.0482	0.0639	0.0529	0.0589	0.0213
8	2	1	1	1	1	0	2	0.0402	0.0433	0.0461	0.0434	0.0422	0.0430	0.0021
9	1	2	0	1	0	1	2	0.0214	0.0231	0.0207	0.0215	0.0210	0.0215	0.0009
10	1	2	1	0	1	0	2	0.0238	0.0214	0.0233	0.0216	0.0222	0.0225	0.0010
11	2	1	0	1	1	1	1	0.0468	0.0847	0.1961	0.0775	0.1849	0.1180	0.0678
12	2	1	1	0	0	1	1	0.2098	0.0750	0.0705	0.2153	0.1217	0.1385	0.0706
13	2	2	0	0	0	0	0	0.1731	0.1632	0.2694	0.1857	0.1961	0.1975	0.0421
14	2	2	1	1	0	0	0	0.1435	0.1504	0.1133	0.1401	0.1433	0.1381	0.0144
15	0	0	0	0	1	1	2	0.0590	0.0583	0.0601	0.0587	0.0572	0.0587	0.0011
16	0	0	1	1	1	1	2	0.0651	0.0656	0.0654	0.0644	0.0666	0.0654	0.0008
17	2	0	0	0	1	0	2	0.0526	0.0605	0.0546	0.0536	0.0576	0.0558	0.0032
18	2	0	1	0	0	1	2	0.0615	0.0618	0.0692	0.0626	0.0587	0.0628	0.0039
19	0	2	0	1	0	0	0	0.0730	0.0599	0.1034	0.0718	0.0931	0.0803	0.0176
20	0	2	1	1	1	0	0	0.0373	0.0516	0.0853	0.0899	0.0516	0.0632	0.0231
21	2	0	0	1	0	1	0	0.1250	0.1255	0.1215	0.1144	0.1155	0.1204	0.0052
22	2	0	1	0	1	0	0	0.2034	0.1961	0.2131	0.2183	0.2273	0.2117	0.0122
23	0	2	0	1	1	1	2	0.0350	0.0345	0.0342	0.0359	0.0336	0.0346	0.0009
24	0	2	1	0	0	1	2	0.0411	0.0436	0.0383	0.0425	0.0408	0.0413	0.0020
25	0	0	0	0	0	0	1	0.0557	0.0586	0.0573	0.0608	0.0566	0.0578	0.0020
26	0	0	1	1	0	0	1	0.0661	0.0658	0.0661	0.0661	0.0626	0.0653	0.0015
27	1	1	0	0	1	1	0	0.2143	0.1601	0.2127	0.1792	0.1247	0.1782	0.0377
28	1	1	1	1	1	1	0	0.2143	0.2026	0.0822	0.1234	0.2183	0.1682	0.0617
29	0	1	0	0	1	0	0	0.0567	0.0536	0.0455	0.0641	0.0464	0.0533	0.0077
30	0	1	1	0	0	1	0	0.0794	0.0945	0.0871	0.0676	0.0786	0.0814	0.0101
31	1	0	0	1	0	0	1	0.0525	0.0529	0.0591	0.0570	0.0556	0.0554	0.0028
32	1	0	1	1	1	0	1	0.0587	0.0583	0.0610	0.0603	0.0611	0.0599	0.0013
33	0	1	0	1	0	1	1	0.0440	0.0449	0.0454	0.0482	0.0453	0.0456	0.0016
34	0	1	1	0	1	0	1	0.0534	0.0514	0.0518	0.0530	0.0530	0.0525	0.0009
35	1	0	0	1	1	1	0	0.0808	0.0977	0.1143	0.1192	0.0940	0.1012	0.0156
36	1	0	1	0	0	1	0	0.0614	0.1121	0.1873	0.2095	0.1848	0.1510	0.0621

Table A.3: 36 run design matrix and simulation results of Graves' method

	Design matrix									Simulation results						
Run	d	m	w	V	N	B	P	C	A	y_{gr1}	y_{gr2}	y_{gr3}	y_{gr4}	y_{gr5}	$\bar{y}_{gr}$	sd_{gr}
1	1	1	0	0	0	0	0	2	1	0.0906	0.0658	0.0494	0.0672	0.1594	0.0865	0.0433
2	1	1	1	1	0	0	0	2	2	0.0708	0.0909	0.1151	0.0455	0.0593	0.0763	0.0273
3	2	2	0	0	1	1	1	1	1	0.2135	0.2132	0.2122	0.2160	0.2165	0.2143	0.0019
4	2	2	1	1	1	1	0	1	2	0.1390	0.1394	0.1372	0.1379	0.1385	0.1384	0.0009
5	1	2	0	0	1	0	0	2	2	0.0901	0.0618	0.1077	0.1119	0.0484	0.0840	0.0280
6	1	2	1	0	0	1	0	2	0	0.0912	0.0589	0.0691	0.1410	0.1771	0.1074	0.0502
7	2	1	0	1	0	0	1	1	2	0.1212	0.1234	0.1212	0.1259	0.1257	0.1235	0.0023
8	2	1	1	1	1	0	1	1	0	0.1183	0.1124	0.0895	0.1042	0.1113	0.1071	0.0111
9	1	2	0	1	0	1	1	1	0	0.0285	0.0815	0.0881	0.1272	0.0468	0.0744	0.0384
10	1	2	1	0	1	0	1	1	1	0.0232	0.0238	0.0287	0.0218	0.0342	0.0263	0.0051
11	2	1	0	1	1	1	0	2	0	0.1212	0.1220	0.1214	0.1567	0.1230	0.1289	0.0155
12	2	1	1	0	0	1	1	2	1	0.2150	0.2289	0.2145	0.2116	0.2131	0.2166	0.0070
13	2	2	0	0	0	0	0	0	2	0.1565	0.2174	0.2167	0.2011	0.1943	0.1972	0.0249
14	2	2	1	1	0	0	0	0	0	0.1386	0.1413	0.1182	0.1379	0.1344	0.1341	0.0092
15	0	0	0	0	1	1	1	2	2	0.0599	0.0596	0.0589	0.0602	0.0583	0.0594	0.0008
16	0	0	1	1	1	1	0	2	0	0.0648	0.0657	0.0667	0.0637	0.0644	0.0650	0.0012
17	2	0	0	0	1	0	0	0	0	0.1116	0.0768	0.0624	0.0597	0.0742	0.0769	0.0207
18	2	0	1	0	0	1	0	0	1	0.1003	0.2114	0.1685	0.0775	0.0887	0.1293	0.0580
19	0	2	0	1	0	0	1	2	0	0.0368	0.0367	0.0370	0.0351	0.0341	0.0359	0.0013
20	0	2	1	1	1	0	1	2	1	0.0416	0.0415	0.0390	0.0415	0.0415	0.0410	0.0011
21	2	0	0	1	0	1	1	2	1	0.1163	0.1162	0.1178	0.1185	0.1158	0.1169	0.0012
22	2	0	1	0	1	0	1	2	2	0.2093	0.2073	0.2074	0.2067	0.2095	0.2080	0.0013
23	0	2	0	1	1	1	0	0	1	0.0349	0.0354	0.0369	0.0367	0.0347	0.0357	0.0010
24	0	2	1	0	0	1	1	0	2	0.0408	0.0413	0.0439	0.0416	0.0432	0.0422	0.0013
25	0	0	0	0	0	0	0	1	0	0.0591	0.0595	0.0591	0.0569	0.0607	0.0591	0.0014
26	0	0	1	1	0	0	0	1	1	0.0666	0.0645	0.0629	0.0662	0.0671	0.0655	0.0018
27	1	1	0	0	1	1	1	0	0	0.0336	0.0327	0.0364	0.0348	0.0368	0.0348	0.0018
28	1	1	1	1	1	1	0	0	1	0.0408	0.0427	0.0393	0.0434	0.0438	0.0420	0.0019
29	0	1	0	0	1	0	0	1	1	0.0452	0.0453	0.0446	0.0453	0.0429	0.0447	0.0010
30	0	1	1	0	0	1	0	1	2	0.0511	0.0507	0.0533	0.0524	0.0534	0.0522	0.0012
31	1	0	0	1	0	0	1	0	1	0.0560	0.0536	0.0567	0.0578	0.0603	0.0569	0.0025
32	1	0	1	1	1	0	1	0	2	0.0590	0.0582	0.0640	0.0610	0.0559	0.0596	0.0031
33	0	1	0	1	0	1	1	0	2	0.0388	0.0463	0.0514	0.0438	0.0484	0.0457	0.0048
34	0	1	1	0	1	0	1	0	0	0.0516	0.0513	0.0539	0.0530	0.0523	0.0524	0.0010
35	1	0	0	1	1	1	0	1	2	0.0618	0.0638	0.0583	0.0643	0.0543	0.0605	0.0042
36	1	0	1	0	0	1	1	1	0	0.0763	0.0665	0.0740	0.0734	0.0659	0.0712	0.0047

Table A.4: 36 run design matrix and simulation results of EE-MH

	Design matrix											Simulation results						
Run	d	m	w	V	N	B	W	K	J	H_1	O	y_{eemh1}	y_{eemh2}	y_{eemh3}	y_{eemh4}	y_{eemh5}	$\bar{y}_{eemh}$	sd_{eemh}
1	1	1	0	0	0	0	2	0	2	1	0	0.0354	0.0362	0.0393	0.0374	0.0374	0.0372	0.0015
2	1	1	1	1	0	0	2	0	2	2	1	0.0494	0.0474	0.0428	0.0464	0.0416	0.0455	0.0032
3	2	2	0	0	1	1	1	1	1	1	1	0.0275	0.0233	0.0226	0.0253	0.0233	0.0244	0.0020
4	2	2	1	1	1	1	1	0	1	2	1	0.0272	0.0268	0.0235	0.0260	0.0316	0.0270	0.0029
5	1	2	0	0	1	0	1	0	2	2	1	0.0219	0.0219	0.0212	0.0231	0.0213	0.0219	0.0008
6	1	2	1	0	0	1	1	0	2	0	0	0.0310	0.0276	0.0232	0.0261	0.0250	0.0266	0.0029
7	2	1	0	1	0	0	2	1	1	2	1	0.0427	0.0429	0.0403	0.0400	0.0451	0.0422	0.0021
8	2	1	1	1	1	0	2	1	1	0	0	0.0399	0.0458	0.0424	0.0468	0.0416	0.0433	0.0029
9	1	2	0	1	0	1	2	1	1	0	0	0.0232	0.0228	0.0226	0.0221	0.0236	0.0229	0.0006
10	1	2	1	0	1	0	2	1	1	1	0	0.0268	0.0278	0.0248	0.0272	0.0261	0.0265	0.0012
11	2	1	0	1	1	1	1	0	2	0	0	0.0364	0.0361	0.0367	0.0364	0.0373	0.0366	0.0005
12	2	1	1	0	0	1	1	1	2	1	1	0.0495	0.0436	0.0464	0.0432	0.0471	0.0460	0.0026
13	2	2	0	0	0	0	0	0	0	2	0	0.1928	0.1301	0.1639	0.2601	0.1944	0.1883	0.0479
14	2	2	1	1	0	0	0	0	0	0	1	0.1253	0.1144	0.1088	0.1650	0.1827	0.1392	0.0328
15	0	0	0	0	1	1	2	1	2	2	1	0.0556	0.0584	0.0592	0.0599	0.0588	0.0584	0.0016
16	0	0	1	1	1	1	2	0	2	0	1	0.0617	0.0623	0.0639	0.0619	0.0619	0.0623	0.0009
17	2	0	0	0	1	0	2	0	0	0	1	0.0552	0.0561	0.0548	0.0540	0.0563	0.0553	0.0010
18	2	0	1	0	0	1	2	0	0	1	0	0.0639	0.0602	0.0612	0.0679	0.0663	0.0639	0.0033
19	0	2	0	1	0	0	0	1	2	0	1	0.0778	0.0838	0.0583	0.0484	0.0663	0.0669	0.0143
20	0	2	1	1	1	0	0	1	2	1	0	0.0554	0.1478	0.0784	0.1002	0.1856	0.1135	0.0528
21	2	0	0	1	0	1	0	1	2	1	0	0.0896	0.0866	0.1866	0.2380	0.1138	0.1429	0.0667
22	2	0	1	0	1	0	0	1	2	2	0	0.0721	0.0969	0.0999	0.1259	0.0959	0.0981	0.0191
23	0	2	0	1	1	1	2	0	0	1	0	0.0361	0.0342	0.0356	0.0345	0.0348	0.0350	0.0008
24	0	2	1	0	0	1	2	1	0	2	1	0.0442	0.0381	0.0378	0.0419	0.0470	0.0418	0.0040
25	0	0	0	0	0	0	1	0	1	0	0	0.0566	0.0619	0.0624	0.0600	0.0597	0.0602	0.0023
26	0	0	1	1	0	0	1	0	1	1	1	0.0681	0.0687	0.0666	0.0624	0.0649	0.0661	0.0026
27	1	1	0	0	1	1	0	1	0	0	1	0.0742	0.0525	0.1386	0.0685	0.1045	0.0877	0.0341
28	1	1	1	1	1	1	0	0	0	1	1	0.0877	0.0879	0.1108	0.0423	0.0751	0.0808	0.0250
29	0	1	0	0	1	0	0	0	1	1	1	0.0611	0.0757	0.0703	0.0576	0.0994	0.0728	0.0165
30	0	1	1	0	0	1	0	0	1	2	0	0.1297	0.1016	0.0889	0.1138	0.1039	0.1076	0.0152
31	1	0	0	1	0	0	1	1	0	1	1	0.0614	0.0609	0.0542	0.0599	0.0566	0.0586	0.0031
32	1	0	1	1	1	0	1	1	0	2	0	0.0661	0.0625	0.0602	0.0603	0.0580	0.0614	0.0030
33	0	1	0	1	0	1	1	1	0	2	0	0.0464	0.0409	0.0472	0.0491	0.0459	0.0459	0.0031
34	0	1	1	0	1	0	1	1	0	0	0	0.0542	0.0516	0.0520	0.0552	0.0525	0.0531	0.0015
35	1	0	0	1	1	1	0	0	1	2	0	0.0964	0.0665	0.1344	0.1058	0.0670	0.0940	0.0286
36	1	0	1	0	0	1	0	1	1	0	1	0.0993	0.0974	0.0991	0.0815	0.1894	0.1133	0.0432

Table A.5: 36 run design matrix and simulation results of EE-Slice

	Design matrix											Simulation results						
Run	d	m	w	V	N	B	E	K	J	H_1	O	y_{eesl1}	y_{eesl2}	y_{eesl3}	y_{eesl4}	y_{eesl5}	$\bar{y}_{eesl}$	sd_{eesl}
1	1	1	0	0	0	0	2	0	2	1	0	0.0355	0.0344	0.0359	0.0361	0.0326	0.0349	0.0014
2	1	1	1	1	0	0	2	0	2	2	1	0.0402	0.0419	0.0423	0.0416	0.0410	0.0414	0.0008
3	2	2	0	0	1	1	1	1	1	1	1	0.0224	0.0225	0.0233	0.0223	0.0222	0.0225	0.0004
4	2	2	1	1	1	1	1	0	1	2	1	0.0254	0.0247	0.0240	0.0223	0.0240	0.0241	0.0011
5	1	2	0	0	1	0	1	0	2	2	1	0.0209	0.0210	0.0216	0.0208	0.0214	0.0212	0.0003
6	1	2	1	0	0	1	1	0	2	0	0	0.0243	0.0222	0.0225	0.0238	0.0271	0.0240	0.0020
7	2	1	0	1	0	0	2	1	1	2	1	0.0375	0.0377	0.0414	0.0371	0.0361	0.0380	0.0020
8	2	1	1	1	1	0	2	1	1	0	0	0.0442	0.0427	0.0446	0.0440	0.0428	0.0437	0.0009
9	1	2	0	1	0	1	2	1	1	0	0	0.0207	0.0206	0.0212	0.0206	0.0197	0.0205	0.0005
10	1	2	1	0	1	0	2	1	1	1	0	0.0227	0.0218	0.0240	0.0226	0.0211	0.0225	0.0011
11	2	1	0	1	1	1	1	0	2	0	0	0.0358	0.0371	0.0381	0.0357	0.0362	0.0366	0.0010
12	2	1	1	0	0	1	1	1	2	1	1	0.0404	0.0428	0.0465	0.0398	0.0426	0.0424	0.0026
13	2	2	0	0	0	0	0	0	0	2	0	0.1364	0.1879	0.1806	0.2870	0.1251	0.1834	0.0640
14	2	2	1	1	0	0	0	0	0	0	1	0.1229	0.1197	0.1541	0.1482	0.1419	0.1374	0.0153
15	0	0	0	0	1	1	2	1	2	2	1	0.0588	0.0591	0.0594	0.0585	0.0579	0.0587	0.0006
16	0	0	1	1	1	1	2	0	2	0	1	0.0650	0.0665	0.0649	0.0644	0.0636	0.0649	0.0011
17	2	0	0	0	1	0	2	0	0	0	1	0.0537	0.0559	0.0553	0.0562	0.0569	0.0556	0.0012
18	2	0	1	0	0	1	2	0	0	1	0	0.0640	0.0596	0.0599	0.0611	0.0583	0.0606	0.0021
19	0	2	0	1	0	0	0	1	2	0	1	0.0518	0.2123	0.2532	0.1768	0.1319	0.1652	0.0776
20	0	2	1	1	1	0	0	1	2	1	0	0.0806	0.0611	0.1354	0.0900	0.0484	0.0831	0.0335
21	2	0	0	1	0	1	0	1	2	1	0	0.1345	0.1012	0.1571	0.0875	0.0921	0.1145	0.0301
22	2	0	1	0	1	0	0	1	2	2	0	0.1910	0.1890	0.0991	0.1231	0.1055	0.1415	0.0451
23	0	2	0	1	1	1	2	0	0	1	0	0.0352	0.0342	0.0344	0.0356	0.0352	0.0349	0.0006
24	0	2	1	0	0	1	2	1	0	2	1	0.0397	0.0411	0.0400	0.0411	0.0404	0.0405	0.0007
25	0	0	0	0	0	0	1	0	1	0	0	0.0569	0.0567	0.0585	0.0584	0.0592	0.0579	0.0011
26	0	0	1	1	0	0	1	0	1	1	1	0.0653	0.0674	0.0677	0.0673	0.0673	0.0670	0.0010
27	1	1	0	0	1	1	0	1	0	0	1	0.0807	0.0693	0.0753	0.1017	0.1766	0.1007	0.0442
28	1	1	1	1	1	1	0	0	0	1	1	0.1535	0.0668	0.0646	0.1665	0.2849	0.1473	0.0903
29	0	1	0	0	1	0	0	0	1	1	1	0.0574	0.0585	0.1050	0.0696	0.0439	0.0669	0.0232
30	0	1	1	0	0	1	0	0	1	2	0	0.0975	0.0939	0.1810	0.0896	0.1099	0.1144	0.0380
31	1	0	0	1	0	0	1	1	0	1	1	0.0558	0.0525	0.0575	0.0551	0.0537	0.0549	0.0019
32	1	0	1	1	1	0	1	1	0	2	0	0.0589	0.0603	0.0595	0.0619	0.0580	0.0597	0.0015
33	0	1	0	1	0	1	1	1	0	2	0	0.0451	0.0448	0.0452	0.0445	0.0452	0.0450	0.0003
34	0	1	1	0	1	0	1	1	0	0	0	0.0521	0.0524	0.0514	0.0524	0.0521	0.0521	0.0004
35	1	0	0	1	1	1	0	0	1	2	0	0.0981	0.1439	0.1003	0.0900	0.2375	0.1340	0.0616
36	1	0	1	0	0	1	0	1	1	0	1	0.2288	0.2660	0.1101	0.1061	0.1191	0.1660	0.0756

Appendix B

Simulation Study Result

Table B.1: 9 run simulation results of MH

	Design matrix		Simulation results						
Run	d	m	y_{mh1}	y_{mh2}	y_{mh3}	y_{mh4}	y_{mh5}	$\bar{y}_{mh}$	sd_{mh}
1	0	0	0.0642	0.0550	0.0627	0.0546	0.0638	0.0601	0.0048
2	1	0	0.0560	0.0544	0.0522	0.0562	0.0574	0.0553	0.0020
3	2	0	0.0541	0.0482	0.0545	0.0625	0.0570	0.0553	0.0051
4	0	1	0.0474	0.0453	0.0466	0.0482	0.0472	0.0469	0.0011
5	1	1	0.0352	0.0363	0.0347	0.0349	0.0346	0.0351	0.0007
6	2	1	0.0382	0.0338	0.0395	0.0371	0.0355	0.0368	0.0022
7	0	2	0.0367	0.0370	0.0367	0.0329	0.0370	0.0361	0.0018
8	1	2	0.0211	0.0210	0.0230	0.0216	0.0206	0.0215	0.0009
9	2	2	0.0308	0.0230	0.0292	0.0452	0.0210	0.0298	0.0095

Table B.2: 9 run simulation results of Slice sampling

	Design matrix		Simulation results						
Run	d	m	y_{sl1}	y_{sl2}	y_{sl3}	y_{sl4}	y_{sl5}	$\bar{y}_{sl}$	sd_{sl}
1	0	0	0.0580	0.0596	0.0584	0.0591	0.0587	0.0588	0.0006
2	1	0	0.0565	0.0563	0.0565	0.0561	0.0544	0.0560	0.0009
3	2	0	0.0573	0.0547	0.0542	0.0549	0.0557	0.0553	0.0012
4	0	1	0.0450	0.0455	0.0451	0.0447	0.0445	0.0450	0.0004
5	1	1	0.0349	0.0348	0.0357	0.0353	0.0338	0.0349	0.0007
6	2	1	0.0379	0.0379	0.0343	0.0353	0.0371	0.0365	0.0016
7	0	2	0.0355	0.0342	0.0347	0.0344	0.0351	0.0348	0.0005
8	1	2	0.0206	0.0205	0.0207	0.0205	0.0210	0.0207	0.0002
9	2	2	0.0260	0.0608	0.0236	0.0287	0.0564	0.0391	0.0180

Table B.3: 9 run simulation results of Graves' method

	Design matrix		Simulation results						
Run	d	m	y_{gr1}	y_{gr2}	y_{gr3}	y_{gr4}	y_{gr5}	$\bar{y}_{gr}$	sd_{gr}
1	0	0	0.0585	0.0576	0.0580	0.0622	0.0602	0.0593	0.0019
2	1	0	0.0552	0.0529	0.0572	0.0556	0.0517	0.0545	0.0022
3	2	0	0.0797	0.1558	0.0788	0.1210	0.0981	0.1067	0.0324
4	0	1	0.0434	0.0465	0.0452	0.0447	0.0451	0.0450	0.0011
5	1	1	0.0366	0.0354	0.0353	0.0368	0.0347	0.0357	0.0009
6	2	1	0.2121	0.1576	0.0779	0.1694	0.1756	0.1585	0.0494
7	0	2	0.0374	0.0334	0.0378	0.0342	0.0335	0.0352	0.0022
8	1	2	0.0207	0.0220	0.0213	0.0236	0.0220	0.0219	0.0011
9	2	2	0.1114	0.1742	0.1649	0.1162	0.1193	0.1372	0.0298

Table B.4: 9 run simulation results of EE-MH

	Design matrix		Simulation results						
Run	d	m	y_{eemh1}	y_{eemh2}	y_{eemh3}	y_{eemh4}	y_{eemh5}	$\bar{y}_{eemh}$	sd_{eemh}
1	0	0	0.0575	0.0603	0.0594	0.0608	0.0609	0.0598	0.0014
2	1	0	0.0543	0.0577	0.0556	0.0575	0.0546	0.0559	0.0016
3	2	0	0.0543	0.0598	0.0551	0.0544	0.0571	0.0561	0.0023
4	0	1	0.0478	0.0477	0.0468	0.0444	0.0475	0.0468	0.0014
5	1	1	0.0371	0.0360	0.0388	0.0372	0.0364	0.0371	0.0011
6	2	1	0.0348	0.0388	0.0346	0.0368	0.0379	0.0366	0.0018
7	0	2	0.0362	0.0363	0.0367	0.0365	0.0373	0.0366	0.0004
8	1	2	0.0208	0.0209	0.0217	0.0220	0.0224	0.0216	0.0007
9	2	2	0.0225	0.0221	0.0225	0.0218	0.0214	0.0221	0.0005

Table B.5: 9 run simulation results of EE-Slice

	Design matrix		Simulation results						
Run	d	m	y_{eesl1}	y_{eesl2}	y_{eesl3}	y_{eesl4}	y_{eesl5}	$\bar{y}_{eesl}$	sd_{eesl}
1	0	0	0.0587	0.0570	0.0584	0.0571	0.0586	0.0579	0.0008
2	1	0	0.0563	0.0554	0.0565	0.0570	0.0539	0.0558	0.0012
3	2	0	0.0560	0.0569	0.0540	0.0545	0.0548	0.0553	0.0012
4	0	1	0.0439	0.0438	0.0441	0.0447	0.0457	0.0444	0.0008
5	1	1	0.0339	0.0349	0.0346	0.0346	0.0331	0.0342	0.0007
6	2	1	0.0355	0.0362	0.0371	0.0352	0.0351	0.0358	0.0008
7	0	2	0.0345	0.0359	0.0344	0.0342	0.0353	0.0349	0.0007
8	1	2	0.0207	0.0212	0.0210	0.0210	0.0205	0.0209	0.0003
9	2	2	0.0226	0.0226	0.0228	0.0295	0.0228	0.0241	0.0031

Table B.6: Simulation time of five algorithms in the $9 - run$ simulation study

Run	MH	Slice	Graves	EE-MH	EE-Slice
1	15s/0.25m	37s/0.62m	16s/0.26m	715 s/11.91m	974 s/16.24m
2	15s/0.25m	40s/0.66m	16s/0.26m	725 s/12.08m	985 s/16.41m
3	15s/0.25m	42s/0.69m	16s/0.26m	732 s/12.20m	993 s/16.55m
4	22s/0.37m	49s/0.82m	23s/0.39m	974 s/16.23m	1247s/20.78m
5	22s/0.37m	54s/0.90m	23s/0.39m	994 s/16.57m	1251s/20.85m
6	22s/0.37m	59s/0.98m	23s/0.39m	999 s/16.65m	1278s/21.30m
7	30s/0.49m	59s/0.98m	31s/0.51m	1236s/20.61m	1509s/25.15m
8	30s/0.49m	69s/1.16m	31s/0.51m	1254s/20.90m	1524s/25.41m
9	30s/0.50m	76s/1.27m	31s/0.51m	1266s/21.09m	1569s/26.15m

Bibliography

Casella, G. and George, E. I. (1992), "Explaining the gibbs sampler," *The American Statistician*, 46, 167–174.

Chib, S. and Greenberg, E. (1995), "Understanding the metropolis-hastings algorithm," *The American Statistician*, 49, 327–335.

Gelman, A., Roberts, G. O., and Gilks, W. R. (1995), "Efficient metropolis jumping rules," *Bayesian Statistics 5*. Oxford University Press.

Geman, S. and Geman, A. (1984), "Stochastic relaxation, gibbs distributions and the bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 721–740.

Graves, T. (2005), "Automatic step size selection in random walk metropolis algorithms," Technical report, Los Alamos National Laboratory.

Hastings, W. K. (1970), "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, 57, 91–109.

Kou, S. C., Zhou, Q., and Wong, W. H. (2006), "Discussion paper equi-energy sampler with applications in statistical inference and statistical mechanics," *The Annals of Statistics*, 34, 1581–1619.

Marinari, E. and Parisi, G. (1992), "Simulated tempering: A new monte carlo scheme," *Europhysics Letters*, 19, 451–458.

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

Miller, A. (2002), *Subset Selection in Regression*, Chapman and Hall/CRC.

Neal, R. M. (2003), "Slice sampling," *The Annals of Statistics*, 31, 705–767.

Robert, C. P. and Casella, G. (1999), *Monte Carlo Statistical Methods*, Springer.

Swendsen, R. H. and Wang, J.-S. (1987), "Nonuniversal critical dynamics in monte carlo simulations," *Physical Review Letters*, 58, 86–88.

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

Wu, C. and Hamada, M. (2000), *Experiments: planning, analysis, and parameter design optimization*, John Wiley and Sons, Inc.

Zhang, Y., Pang, S., and Wang, Y. (2001), "Orthogonal arrays obtained by generalized hadamard product," *Discrete Mathematics*, 238, 151–170.