Convergence and Efficiency of Adaptive MCMC

by

Jinyoung Yang

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Statistical Sciences
University of Toronto

Abstract

Convergence and Efficiency of Adaptive MCMC

Jinyoung Yang

Doctor of Philosophy

Graduate Department of Statistical Sciences

University of Toronto

2016

Adaptive Markov Chain Monte Carlo (MCMC) algorithms attempt to 'learn' from the results of past iterations so the Markov chain can converge quicker. Unfortunately, adaptive MCMC algorithms are no longer Markovian, so their convergence is difficult to guarantee.

The first part of this thesis approaches the problem via finite adaption. We develop new diagnostics to determine whether the adaption is still improving the convergence. We present an algorithm which automatically stops adapting once it determines further adaption will not increase the convergence speed. Our algorithm allows the computer to tune a 'good' Markov chain through multiple phases of adaption, and then run conventional non-adaptive MCMC. In this way, the efficiency gains of adaptive MCMC can be obtained while still ensuring convergence to the target distribution.

The second part of the thesis proves convergence to stationarity of adaptive MCMC algorithms, assuming only simple easily-verifiable upper and lower bounds on transition densities. In particular, the transition and proposal densities are not required to be continuous, thus improving on the previous ergodicity results of Craiu et al. (2015).

The third part of the thesis develops an adaptive algorithm which can locally adjust to an irregularly-shaped target distribution. When the target distribution of a Markov chain is irregularly shaped, a 'good' proposal distribution for one part of the state space might be a 'poor' one for another part of the state space. We consider a component-wise multiple-try

Metropolis (CMTM) algorithm that can automatically choose a better proposal out of a set of proposals from different distributions. The computational efficiency is increased using an adaption rule for the CMTM algorithm that dynamically builds a better set of proposal distributions as the Markov chain runs. We also demonstrated theoretically the ergodicity of the adaptive chain.

Acknowledgements

I would like to thank my thesis supervisor Professor Jeffrey Rosenthal for his support and guidance during my Ph.D. study. His extensive knowledge, energy, enthusiasm and patience helped me greatly in completing my Ph.D. thesis. He has been a tremendous mentor for me and I was lucky to have him as my supervisor.

I also would like to thank my co-supervisor Professor Radu Craiu. I am truly grateful for his encouragement, enthusiasm, brilliant ideas, and most importantly all the support he gave me.

I wish to thank my thesis committee member Professor Mike Evans for offering helpful insights and for teaching me so much in my early year of Ph.D. study. I thank Professor Neal Madras and Professor Radford Neal for spending hours reading my thesis and providing valuable comments to improve it. I also thank Professor Peter Rosenthal for his advice on the math problem I encountered.

Thank you to Andrea Carter, Christine Bulguryemez, Annette Courtemanche, and Angela Fleury for helping me with my administrative problems. Also, thank you to Dermot Whelan for the help on the computing problems.

I also wish to thank my friends and colleges at the University of Toronto. You made my life much more enjoyable.

Finally, a special thanks to my family, who always have given me unconditional love and support throughout my entire life. Thank you for always being there for me.

Contents

1 Introduction **1**

2 Adaptive MCMC **5**

2.1 Why MCMC? 5

2.2 Markov Chain Theory 6

2.3 Metropolis-Hastings Algorithm 8

2.3.1 Optimal Acceptance Rate 9

2.4 Some Adaptive MCMC Algorithms 11

2.4.1 Adaptive Metropolis Algorithm 11

2.4.2 Single Component Adaptive Metropolis Algorithm 12

2.4.3 Adaptive Metropolis-within-Gibbs Algorithm 13

2.5 Convergence of Adaptive MCMC 14

3 Automatically Tuned General-Purpose MCMC via New Adaptive Diagnostics **16**

3.1 Introduction 16

3.2 Approach 18

3.3 Background 22

3.4 Technical Details 24

3.4.1 1^{st} Adaption Phase . . . 26
3.4.2 Transient Phase . . . 27
3.4.3 2^{nd} Adaption Phase . . . 29
3.4.4 Sampling Phase . . . 29
3.4.5 Extension for Strongly Multimodal Targets . . . 30
3.5 Applications . . . 35
3.5.1 Multivariate Normal Distribution . . . 36
3.5.2 Logistic Regression . . . 39
3.5.3 Pump Failure Data . . . 42
3.5.4 Variance Components Model (VCM) . . . 47
3.5.5 A Strongly Multimodal Example . . . 53
3.6 Discussion . . . 57
3.6.1 Acceptance Rate . . . 57
3.6.2 Significance of Transient and 2^{nd} Adaption Phase . . . 58
3.6.3 Comparison with a Full-dimensional Metropolis . . . 59
3.6.4 Initial Value for a Markov Chain . . . 61

4 Ergodicity of Combocontinuous Adaptive MCMC Algorithms 63

4.1 Introduction . . . 63
4.2 Background about Adaptive MCMC . . . 65
4.3 Combocontinuous Functions . . . 66
4.4 The Bounded Adaption Metropolis (BAM) Algorithm . . . 68
4.5 More General Conditions . . . 71
4.6 Main Result . . . 76
4.7 Generalisation of Dini's Theorem . . . 77
4.8 Lemmas About Combocontinuity . . . 79

4.9 Proof of Theorem 7 . . . 89
4.10 Numerical examples . . . 90
4.10.1 Application: 9-dimensional Multivariate Normal Distribution . . . 90
4.10.2 Application: Pump Failure Model . . . 91

5 Adaptive Component-wise Multiple-Try Metropolis Sampling 95

5.1 Introduction . . . 95
5.2 Component-wise Multiple-Try Metropolis . . . 98
5.2.1 Algorithm . . . 98
5.2.2 Optimal α . . . 101
5.3 Adaptive Component-wise Multiple-Try Metropolis . . . 104
5.3.1 CMTM Favours Locally 'Better' Proposal Distributions . . . 104
5.3.2 Comparison with a Mixture Transition Kernel . . . 106
5.3.3 The Adaptive CMTM . . . 109
5.3.4 To Adapt or Not To Adapt? . . . 111
5.3.5 Convergence of Adaptive CMTM . . . 113
5.4 Applications . . . 114
5.4.1 Variance Components Model . . . 114
5.4.2 "Banana-shaped" Distribution . . . 117
5.4.3 Orange Tree Growth Data . . . 119
5.5 Comparsion of Adaptive Algorithms . . . 123
5.6 Conclusion and Discussion . . . 128

6 Conclusion 131

Bibliography 133

List of Tables

3.1 Results of MCMC: Multivariate Normal. Results of 10 independent runs of full algorithm in Section 3.4. 37

3.2 Summary Statistics for the Estimates in Table 3.1: Multivariate Normal. μ is the true mean of the target distribution. 38

3.3 Results of MCMC:Logistic Regression. Results of 10 independent runs of full algorithm in Section 3.4. 41

3.4 Summary Statistics for the Estimates in Table 3.3: Logistic Regression. Results of a run from R package 'mcmc' and GLM estimates are also presented, for comparison. 41

3.5 Pump Failure Data 42

3.6 Results of MCMC: Pump Failure Data. Results of 10 independent runs of full algorithm in Section 3.4. 44

3.7 Summary Statistics for the Estimates in Table 3.6: Pump Failure Data. Results from the OpenBUGS website, obtained via Gibbs sampler, are also presented, for comparison. 45

3.8 Results of MCMC:VCM, flat inverse gamma priors. Results of 10 independent runs of full algorithm in Section 3.4. 48

3.9 Summary Statistics for the Estimates in Table 3.8: VCM, flat inverse gamma priors. Results from a Gibbs sampler run are also presented, for comparison. 49

3.10 Results of MCMC:VCM, concentrated inverse gamma priors. Results of 10 independent runs of full algorithm in Section 3.4. 51
3.11 Summary Statistics for Estimates in Table 3.10: VCM, concentrated inverse gamma priors. Results from Gibbs sampler are also presented, for comparison. 52
3.12 Results of MCMC: Multimodal Distribution. Results of 10 independent runs of full algorithm in Section 3.4.5. 55
3.13 Summary Statistics for the Estimates in Table 3.12: Multimodal Distribution. μ is the true mean of target distribution. 55

4.1 Pump Failure Data 92

5.1 Proportion of selected proposals 105
5.2 Acceptance rate of selected proposals 105
5.3 CMTM. Proportion of proposal distribution selected 107
5.4 Acceptance rate on selected proposals 108
5.5 Performance comparison (averaged over 100 runs) 108
5.6 Ending $\sigma_{k,j}$'s 111
5.7 Performance comparisons (averaged over 100 runs) 112
5.8 Proportion of proposal distribution selected 112
5.9 Acceptance rate on selected proposals 112
5.10 Dyestuff Batch Yield (in grams) 115
5.11 $\sigma_{k,j}$'s used in comparing the standard CMTM and the standard component-wise Metropolis. Variance components model 116
5.12 Growth of Orange Trees 120
5.13 $\sigma_{k,j}$'s used in comparing the standard CMTM and the standard component-wise Metropolis. Orange tree growth data 122

List of Figures

3.1 Algorithm flowchart 19

3.2 Trace Plots for Multivariate Normal 39

3.3 Trace Plots for Logistic Regression. Each coordinate corresponds to the parameters listed in Table 3.3 as ordered. 42

3.4 Trace Plots for Pump Failure Data. Each coordinate corresponds to the parameters listed in Table 3.6 as ordered. 46

3.5 Trace Plots for Variance Components Model (with flat inverse gamma priors). Each coordinate corresponds to the parameters listed in Table 3.8 as ordered. 50

3.6 Trace Plots for Variance Components Model (with concentrated inverse gamma priors). Each coordinate corresponds to the parameter listed in Table 3.10 as ordered. 53

3.7 Trace Plots for Mixture of Three 3-Dimensional Multivariate Normals. Each row represents each coordinate and each column represents each chain trapped in different mode until the sampling phase. 56

4.1 Two ways of truncating a normal density: with (a) a "firm" truncation (left), or (b) a "linear" truncation (right). 67

4.2 Trace plots (of coordinate 7) for a Bounded Adaption Metropolis (left) versus a Standard Metropolis algorithm with proposal kernel $N(X_n, I_d)$ (right), on a 9-dimensional multivariate normal target distribution, showing the superiority of the BAM algorithm. 91

4.3 Trace plots (of coordinate 10) for the Pump Failure Model example for a Bounded Adaption Metropolis algorithm (top left), compared to Standard Metropolis algorithms with proposal distributions whose Gaussian covariance matrices are the d-dimensional identity (top right), 0.01 times this identity (bottom left), and 0.001 times this identity (bottom right). 93

5.1 Target density plot. 2-dimensional mixture of two normals 101

5.2 Proportion of proposal distribution selected. Coordinate 1: Red, Blue, Green, Orange and Purple lines show behaviour when $\sigma_{k,j} = 1, 2, 4, 8, 16$, respectively. 102

5.3 Two-Dimensional Mixture of two Gaussians: Mean squared jumping distance vs. α for one run (left panel) and averaged over 100 runs (right panel). . . . 103

5.4 4-Dimensional Mixture of two Gaussians: Means squared jumping distance vs. α for one run (left panel) and averaged over 100 runs (right panel). . . . 104

5.5 Adaptive CMTM vs. non-adaptive CMTM. Variance components model. The red represents the adaptive CMTM runs and the blue represents the non-adaptive CMTM runs. ESS is calculated after averaging ACT over 50 independently replicated runs. 116

5.6 Standard (non-adaptive) CMTM vs. standard component-wise Metropolis. Variance components model. The red represents the CMTM runs and the blue represents the component-wise Metropolis runs. ESS is calculated after averaging ACT over 100 replicative runs. 117

5.7 Adaptive CMTM vs. non-adaptive CMTM. "Banana-shaped" distribution. The red represents the adaptive CMTM runs and the blue represents the non-adaptive CMTM runs. ESS is calculated after averaging ACT over 50 replicative runs. 118

5.8 Standard (non-adaptive) CMTM vs. standard component-wise Metropolis. "Banana-shaped" distribution. The red represents the CMTM runs and the blue represents the component-wise Metropolis runs. ESS is calculated after averaging ACT over 50 replicative runs. 119

5.9 Adaptive CMTM vs. non-adaptive CMTM. Orange tree growth data. The red represents the adaptive CMTM runs and the blue represents the non-adaptive CMTM runs. ESS is calculated after averaging ACT over 50 replicative runs. 121

5.10 Standard (non-adaptive) CMTM vs. standard component-wise Metropolis. Orange tree growth data. The red represents the CMTM runs and the blue represents the component-wise Metropolis runs. ESS is calculated after averaging ACT over 100 replicative runs 123

5.11 Comparison of ESS for different adaptive schemes. The red represents the ACMTM algorithm; the purple represents the AMwG algorithm; the green represents the SCAM algorithm; and the orange represents the AM algorithm. In each row, the right panel is a close up that identifies features that cannot be clearly seen in the left panel. ESS is calculated after averaging ACT over 50 independently replicated runs. 126

5.12 Comparison of ESS/CPUtime for different adaptive schemes. The red represents the ACMTM algorithm; the purple represents the AMwG algorithm; the green represents the SCAM algorithm; and the orange represents the AM algorithm. In each row, the right panel is a close up that identifies features that cannot be clearly seen in the left panel. ESS is calculated after averaging ACT over 50 independently replicated runs. 127

Chapter 1

Introduction

Monte Carlo methods, Markov Chain Monte Carlo (MCMC) methods provide a powerful tool to analyze complex probability distributions. (Madras 2000, 2002; Brooks et al. 2011). The development of MCMC has freed researchers to build complex statistical models if necessary. MCMC methods are widely used in various fields such as computational physics, computational biology and computational linguistics, to name a few. MCMC also helped in the growth of Bayesian inference by introducing a means to analyze posterior distributions derived from complex hierarchical models.

One of the main issue in MCMC algorithms is their speed of convergence. MCMC methods relies on the Markov chain convergence theorem, which guarantees the eventual convergence of the Markov chain to the target distribution, the distribution in interest. The time and resource constraint in practice requires more than the eventual convergence of the algorithm to the target distribution. One way to improve the efficiency of MCMC algorithms is adaptive MCMC methods. It is a method that modifies the Markov chain 'on the fly' so the Markov chain can achieve quicker convergence. The rationale behind an idea of adaptive method is: First, the observation that some Markov chains converges quicker than others for a specific target distribution. Second, the history of Markov chain provide some information

with regards to what would be 'better' Markov chain for the particular target distribution. This makes sense intuitively. If a Markov chain proposes a new value at each iteration drawn from a distribution that resembles certain characteristics of the target distribution (e.g., covariance structure), the chain is going to converge to the target distribution quicker than the one which proposes a new value from a distribution that is a lot different with the target distribution. And since Markov chain converges to target distribution, as the chain runs, the collection of the past and current states becomes more like a sample from a target distribution. The adaptive MCMC methods have been extensively studied by many researchers, and it has been shown that the adaptive MCMC algorithms can significantly improve the efficiency of an MCMC algorithm. (e.g. Haario et al. 2001, 2006; Roberts and Rosenthal 2009; Giordani and Kohn 2010; Vihola 2012; Turro et al. 2007)

Many adaptive MCMC algorithms use historical information obtained during the run to modify the chain. This destroys the 'Markov' property of the chain, so the conventional Markov chain convergence theorem cannot be applied. Proving the convergence of an adaptive algorithm has been a challenge. Many have attempted and proved the convergence of an adaptive algorithm (e.g. Haario et al. 2001, 2006; Giordani and Kohn 2010; Vihola 2012; Atchadé and Rosenthal. 2005; Andrieu and Moulines 2006; Andrieu and Atchadé 2007; Roberts and Rosenthal 2007; Fort et al. 2011), but the conditions they presented to assure the convergence are not always easy to verify in practice. There have been efforts to develop more easily checkable conditions to ensure the convergence of an adaptive MCMC algorithm.

In Roberts and Rosenthal (2007), it was shown that any adaptive MCMC algorithm converges to its target distribution if it satisfies two conditions: Diminishing Adaption, which suggests the algorithm adapts less and less as the chain moves along, and Containment, which suggests the convergence time of the algorithm is bounded in probability. Diminishing Adaption is easily achievable through the stage of the chain construction, but Containment is in many instances hard to verify. Craiu et al. (2015) shows the efforts to develop easy-to-

verify condition for Containment, and in return easy-to-apply conditions to show a particular adaptive MCMC algorithm is guaranteed to converge.

In this thesis, we develop a couple of MCMC algorithms which are demonstrated to be more efficient than pre-existing algorithms on several MCMC examples. We also make a contribution on developing general conditions to guarantee the convergence of an adaptive MCMC algorithm.

In Chapter 2 of this thesis, we introduces the MCMC and adaptive MCMC theories and algorithms, which gives the background knowledge to understand the findings presented in the following chapters.

In Chapter 3 (available as a separate paper in Yang and Rosenthal (2016)), we develop a finitely adapting algorithm, which can be generally applied to most existing MCMC problems. The motivation behind this algorithm is as follows. We want the possible efficiency improvement from an adaptive MCMC algorithm. However, verifying the convergence of an adaptive MCMC algorithm is not always an easy task, and the conditions found by the researchers to ensure convergence are more restrictive than conditions for standard MCMC algorithms. Our approach to this problem is to develop adaptive diagnostics which detects if the chain has gained most of benefit from adaption so we know when to stop the adaption and take the chain parameters from the adaption. Essentially, we stop the adaption once our diagnostics says the chain is now 'tuned', and we run the standard MCMC algorithm after with those tuned parameters. As a result, we know our algorithm will converge to its target distribution with an improved convergence speed through tuning from the adaption.

In Chapter 4 (available as a separate paper in Rosenthal and Yang (2016)), we extend the findings from Roberts and Rosenthal (2007) to make the conditions which guarantee the convergence of an adaptive MCMC algorithm more general. We first define the concept of 'combocontinuous' function, which is a generalization of piecewise-continuous function. Continuity of the transition kernel densities is one of the conditions Roberts and Rosenthal

(2007) imposes for an adaptive MCMC algorithm to be guaranteed to converge, which is somewhat restrictive. Thus, we relax the continuity assumption of the transition densities to 'combocontinuity' so we can easily verify the convergence of wider range of adaptive MCMC algorithms.

In Chapter 5 (available as a separate paper in Yang et al. (2016)), we develop an adaptive algorithm for the component-wise multiple-try Metropolis (CMTM) algorithm. Our motivation starts from the concern that many target distributions have irregular shape, thus a Markov chain that works well in one part of the state space might not work so well for another part of the state space as the shapes of the target distribution in these areas are quite different. We applied a component-wise multiple-try Metropolis algorithm(CMTM) to solve this problem. We found that when the CMTM selects a transition kernel out of multiple kernels available, it accounts for the local shape of the target distribution around the current state. We then develop an adaptive algorithm for the CMTM to provide the algorithm a good set of transition kernels it can choose from. We also prove the convergence of the adaptive CMTM algorithm.

We conclude this thesis in Chapter 6.

Chapter 2

Adaptive MCMC

2.1 Why MCMC?

Markov Chain Monte Carlo (MCMC) methods are widely used to analyze complex probability distributions. We show here in a simple way how MCMC helps in this regards.

Let's say there is a random variable X, which has a distribution function G defined on a state space $\mathcal{X}$. Let G_u be an unnormalized version of G and g_u be the density function for G_u. Assume $0 < \int_{\mathcal{X}} g_u(x)dx < \infty$. Suppose we want to know the value of $E[f(X)]$ for some function $f : \mathcal{X} \to \mathbb{R}$. To calculate $E[f(X)]$, we have to calculate the integral

$$E[f(X)] = \frac{\int_{\mathcal{X}} f(x)g_u(x)dx}{\int_{\mathcal{X}} g_u(x)dx}$$

It is not always possible to find a value of some integral. An alternative way to find (approximately) the value of $E[f(X)]$ is calculating $\frac{1}{M}\sum_i f(X_i)$, where a large number, M, of X_i are drawn from $X_i \sim G(X)$. Then, by the law of large numbers, for a large M,

$$E[f(X)] \approx \frac{1}{M}\sum_i f(X_i).$$

Still, there exists another problem. For many distribution functions, it is not possible to directly sample from them. MCMC algorithm takes advantage of the property of Markov chain convergence to sample from any distribution G. Thus, even with a complex Bayesian hierarchical model with a complex posterior function, we can calculate necessary functional values via MCMC algorithms.

We next introduce some definitions and theorems related to Markov chains, which have been foundations to build MCMC methods.

2.2 Markov Chain Theory

A sequence of random variables $X_0, X_1, \ldots,$ is a Markov chain if

$$\mathbf{P}[X_{n+1} \in A \,|X_0, X_1, \ldots, X_n] = \mathbf{P}[X_{n+1} \in A \,|X_n], \qquad \forall A\,.$$

A Markov chain has three components: state space, initial distribution, and transition probability. State space $\mathcal{X}$ is the collection of possible values a random variable X_i can take. Initial distribution is the marginal distribution of X_0. Transition probability distribution (transition kernel) is the conditional distribution of X_{n+1} given X_n, i.e. $P(x, y) = \mathbf{P}[X_{n+1} = y|X_n = x]$ for all $x, y \in \mathcal{X}$. (Note that we denote a k-step transition probability as $P^k(x, y) = \mathbf{P}[X_{n+k} = y|X_n = x]$.)

Next, we introduce some of the definitions and theorems of Markov chains, which are essential to understand MCMC algorithms. Let $\mathcal{X}$ be the state space with a corresponding Borel σ-algebra $\mathcal{F}$.

Definition 1. *A probability distribution π defined on $(\mathcal{X}, \mathcal{F})$ is a stationary distribution for*

a Markov chain on $(\mathcal{X}, \mathcal{F})$ if

$$\int_{\mathcal{X}} \pi(dx)P(x, A) = \pi(A)\,, \qquad \forall A \in \mathcal{F}$$

Definition 2. *A Markov chain defined on $(\mathcal{X}, \mathcal{F})$ is reversible with respect to some probability distribution π also defined on $(\mathcal{X}, \mathcal{F})$ if*

$$\pi(dx)P(x, dy) = \pi(dy)P(y, dx)\,, \qquad \forall x, y \in \mathcal{X}\,.$$

Theorem 1. *Consider a Markov chain that is defined on $(\mathcal{X}, \mathcal{F})$. If the Markov chain is reversible with respect to some probability distribution π also defined on $(\mathcal{X}, \mathcal{F})$, then π is a stationary distribution for the Markov chain.*

Proof. See e.g. Roberts and Rosenthal (2004). □

Definition 3. *There exist a nonzero, σ-finite measure ψ on $(\mathcal{X}, \mathcal{F})$. A Markov chain on $(\mathcal{X}, \mathcal{F})$ is ϕ-irreducible if*

$$\mathbf{P}(\tau_A < \infty | X_0 = x) > 0\,, \qquad \forall x \in \mathcal{X},\ \ \forall A \in \mathcal{F} \text{ with } \psi(A) > 0\,,$$

where $\tau_A = \inf\{n \geq 1 : X_n \in A\}$.

Definition 4. *A Markov chain defined on $(\mathcal{X}, \mathcal{F})$ is aperiodic if there does not exist disjoint subsets $\mathcal{X}_1, \mathcal{X}_2, \ldots \mathcal{X}_d \subset \mathcal{X}$ with $\cup_{i=1}^{d} \mathcal{X}_i = \mathcal{X}$, $d \geq 2$, such that $P(x, \mathcal{X}_{i+1}) = 1$ for $\forall x \in \mathcal{X}_i$ and $P(x, \mathcal{X}_1) = 1$ for $\forall x \in \mathcal{X}_d$.*

Definition 5. *The total variation distance between two probability measures $\mu(\cdot)$ and $\nu(\cdot)$ on $(\mathcal{X}, \mathcal{F})$ is*

$$\|\mu(\cdot) - \nu(\cdot)\| = \sup_{A \in \mathcal{F}} |\mu(A) - \nu(A)|\,.$$

Definition 6. *A transition kernel P is ergodic for $\pi(\cdot)$ on $(\mathcal{X}, \mathcal{F})$ if for all $x \in \mathcal{X}$,*

$$\lim_{n\to\infty} \|P^n(x,\cdot) - \pi(\cdot)\| = 0\,.$$

Theorem 2 (Markov Chain Convergence Theorem)**.** *Consider an aperiodic, ϕ-irreducible Markov chain defined on $(\mathcal{X}, \mathcal{F})$, which has a stationary distribution π. Then, for π-almost every $x \in \mathcal{X}$, the Markov chain converges to π in total variation distance, i.e.*

$$\lim_{n\to\infty} \|P^n(x,\cdot) - \pi(\cdot)\| = 0\,.$$

Proof. See e.g. Meyn and Tweedie (1993). □

2.3 Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm (Metropolis et al. 1953; Hastings 1970) is one of the most well-known, popular algorithms in MCMC. It works as follows.

Let $X_1, X_2, ...$ be the states of a Markov chain defined on a state space $\mathcal{X}$. Let π_u be the possibly unnormalized target density on $\mathcal{X}$. At each iteration n, a new value Y_{n+1} is drawn from a proposal distribution $Q(X_n, \cdot)$. The new proposal Y_{n+1} is accepted with a probability $a(X_n, Y_{n+1})$ defined as

$$a(x,y) = \min\big(1, \frac{\pi_u(y)q(y,x)}{\pi_u(x)q(x,y)}\big)\,,$$

where q is a density of the proposal distribution Q. If the new proposal Y_{n+1} is accepted, then $X_{n+1} = Y_{n+1}$. If Y_{n+1} is rejected, then $X_{n+1} = X_n$.

The convergence of the Metropolis-Hastings algorithm can be proved through the reversibility (i.e. $\pi(dx)P(x,dy) = \pi(dy)P(y,dx)$).

Theorem 3. *A ϕ-irreducible, full-dimensional Metropolis-Hastings algorithm converges to the target distribution π in total variation distance.*

Instead of updating all coordinates at once as in the full-dimensional Metropolis-Hastings algorithm, one can update only one coordinate at a time. This is called the component-wise Metropolis Hastings (CMH) algorithm, or sometimes called Metropolis-within-Gibbs (MwG) algorithm(Metropolis et al. 1953; Tierney 1994). The CMH algorithm proposes a new value for the next state by updating only one coordinate at a time. This can improve the computational efficiency in high dimension compared to a full-dimensional Metropolis-Hastings algorithm.

Let $X_0, X_1, \ldots, X_n$ be a d-dimensional Markov chain. Draw $Y_{n+1,j}$ from the proposal distribution $Q(X_n, \cdot)$. The new update $(X_{n,1}, \ldots, X_{n,j-1},$
$Y_{n+1,j}, X_{n,j+1} \ldots, X_{n,d})$ is accepted with a probability

$$a_j = \min(1, \frac{q(X_{n.j}, Y_{n+1,j})\pi_u(X_{n,1}, \ldots, X_{n,j-1}, Y_{n+1,j}, X_{n,j+1} \ldots, X_{n,d})}{q(X_{n.j}, Y_{n+1,j})\pi_u(X_{n,1}, \ldots, X_{n,j-1}, X_{n,j}, X_{n,j+1} \ldots, X_{n,d})})$$

The CMH algorithm can update coordinates sequentially, from $j = 1$ to $j = d$ and repeat from $j = 1$ again, or it can randomly choose a coordinate j to update each iteration.

2.3.1 Optimal Acceptance Rate

The Metropolis-Hastings algorithm accepts or rejects a new proposal at every iteration. We can calculate the acceptance rate for the algorithm, which is the rate at each iteration a new proposal is accepted. Intuitively, if the rate is too low, this implies new proposals are barely accepted, and the chain is stuck at the same value for a long period of time. This is not ideal for a Markov chain to efficiently search through the state space. If the acceptance rate is too high, this implies each proposal only moves from the current state in a small magnitude,

and even though the acceptance rate is high, the chain overall does not move through the state space quickly.

Roberts et al. (1997) theoretically proved the optimal acceptance rate for a d-dimensional random walk Metropolis algorithm on $\mathbb{R}^d$ with a Gaussian proposal distribution $N(X_n, \sigma^2 I_d)$ and with a target density function in the form of

$$\pi(x) = \prod_{i=1}^{d} f(x_i) \tag{2.1}$$

is 0.234 as $d \to \infty$.

Roberts and Rosenthal (2001) extended the findings of Roberts et al. (1997) and showed that the asymptotic optimal acceptance rate is still 0.234 if the form of the target density function is

$$\pi(x) = \prod_{i=1}^{d} C_i f(C_i x_i) \tag{2.2}$$

in which $\{C_i > 0\}$ are are arbitrary scaling factors. $\{C_i\}$ are assumed to be i.i.d. positive random variables from some distribution.

Numerical studies also showed that the optimal acceptance rate 0.234 is quite robust since it holds for d as low as 5 (Gelman et al. 1996; Roberts and Rosenthal 2001) and for the target density not exactly in the form of (2.1) or (2.2) (Roberts and Rosenthal 2001). Gelman et al. (1996) and Roberts and Rosenthal (2001) also showed through simulation that for one dimensional random walk Metropolis algorithm with normal jumping kernel and standard normal target distribution, the asymptotic optimal acceptance rate is 0.44 approximately.

Haario et al. (2001) suggested that to increase the efficiency of the random walk Metropolis algorithm with Gaussian proposal distribution, let the proposal kernel to be $N(X_n, c\Sigma_p)$ where Σ_p is the covariance matrix of the target distribution. $c = 2.38^2/d$ is shown to yield

the optimal acceptance rate of 0.234 under certain conditions (Roberts et al. 1997).

2.4 Some Adaptive MCMC Algorithms

2.4.1 Adaptive Metropolis Algorithm

Haario et al. (2001) introduced the Adaptive Metropolis (AM) algorithm. The core idea under the adaption scheme of the AM algorithm is to speed up the convergence by tuning the covariance matrix of the proposal distribution so it resembles that of the target distribution.

Let X_0 be the initial state and $X_0, X_1, \ldots, X_n$ be the states already sampled. Then a new proposal Y_{n+1} is drawn from a Gaussian proposal distribution $N(X_n, c\Sigma_{n+1})$. Y_{n+1} is accepted the same way as the Metropolis-Hastings algorithm with a probability

$$a(X_n, Y_{n+1}) = \min(1, \frac{\pi_u(Y_{n+1})}{\pi_u(X_n)}) .$$

Note that since the proposal distribution is symmetric, $q(X_n, Y_{n+1})$ and $q(Y_{n+1}, X_n)$ are canceled out .

Σ_{n+1} is found by

$$\Sigma_{n+1} = \begin{cases} \Sigma_0, & n \leq n_0 \\ Cov(X_0, \ldots, X_n) + \epsilon I_d, & n > n_0 \end{cases}$$

where Cov is the empirical covariance function. Σ_0 is the initial covariance matrix based on the best prior knowledge from the users, and n_0 is a burn-in period to collect the enough sample to calculate the first empirical covariance matrix. ϵI_d is to prevent the covariance matrix from becoming singular. And the constant c is the scaling parameter and assume the value $c = (2.38)^2/d$ from Gelman et al. (1996).

Haario et al. (2001) showed the recursion formula for Σ_n,

$$\Sigma_{n+1} = \frac{n-1}{n}\Sigma_n + \frac{1}{n}(n\overline{X}_n\overline{X}_n^T - (n+1)\overline{X}_{n+1}\overline{X}_{n+1}^T + X_nX_n^T + \epsilon I_d)\,,$$

so the user can save some computational cost to update the empirical covariance matrix Σ_n at every iteration. (It is easy to see that $\overline{X}_n = \frac{1}{n}\sum_{i=0}^{n-1} X_i$ can also be updated iteratively.)

2.4.2 Single Component Adaptive Metropolis Algorithm

The Single Component Adaptive Metropolis (SCAM) algorithm was introduced by Haario et al. (2005). It has the same intuition as the AM algorithm, but this time it finds the empirical variance for each coordinate rather than the empirical covariance matrix for full-dimension. Basically, it runs the AM algorithm on each coordinate separately. A new state X_{n+1} is updated coordinate-by-coordinate, and once every coordinate is updated then X_{n+1} is considered to be updated.

Draw $Y_{n+1,j}$ from the proposal distribution $N(X_{n,j}, sv_{n,j})$. The new update $(X_{n+1,1}, \ldots, X_{n+1,j-1}, Y_{n+1,j}, X_{n,j+1} \ldots, X_{n,d})$ is accepted with a probability

$$a_j = \min(1, \frac{\pi_u(X_{n+1,1}, \ldots, X_{n+1,j-1}, Y_{n+1,j}, X_{n,j+1} \ldots, X_{n,d})}{\pi_u(X_{n+1,1}, \ldots, X_{n+1,j-1}, X_{n,j}, X_{n,j+1} \ldots, X_{n,d})})\,.$$

$v_{n,j}$ is determined by

$$v_{n,j} = \begin{cases} v_{0,j}, & n \le n_0 \\ Var(X_{0,j}, \ldots, X_{n,j}) + \epsilon, & n > n_0 \end{cases}$$

where Var is the empirical variance function. $v_{0,j}$ is the initial variance of the proposal distribution for the coordinate j, which is chosen by the best prior knowledge from the users.

n_0 is a burn-in period, as in the AM algorithm, and ϵ this time is to prevent the variance from becoming zero. Haario et al. (2005) set the constant $s = 2.38^2$ based on Gelman et al. (1996). Haario et al. (2005) also provided a recursive formula for the variance to save the computational cost. If one denotes $g_{n+1} = Var(x_0, \ldots, x_n)$, then the recursive formula for the variance is

$$g_{n+1} = \frac{n-1}{n} g_n + \overline{x}_n^2 + \frac{1}{n} x_n^2 - \frac{n+1}{n} \overline{x}_{n+1}^2.$$

where $\overline{x}_n = \frac{1}{n} \sum_{i=0}^{n-1} x_i$,

2.4.3 Adaptive Metropolis-within-Gibbs Algorithm

The Adaptive Metropolis-within-Gibbs (AMwG) algorithm is introduced in Roberts and Rosenthal (2009). The algorithm is built based on the idea that there is an 'optimal' acceptance rate for a random walk Metropolis algorithm, which yields 'optimal' mixing of the chain.

The AMwG algorithm updates X_n component-by-component, and the proposal distribution for each coordinate is $N(X_{n,j}, \sigma_{n,j}^2)$, where j indexes for coordinate. It modifies the the proposal variance $\sigma_{n,j}^2$ based on the acceptance rate of the proposals for the chain.

Let ls_j be the logarithm of $\sigma_{n,j}$. Set the initial value of $ls_j = 0$ for all j. ls_j for each coordinate is adjusted upward or downward by $\delta(n)$ after each of n^{th} "batch" of 50 iterations. The adjustment is based on the acceptance rate of the last n^{th} batch. The adaption mechanism tries to achieve the 'optimal' acceptance rate of 0.44 for one dimensional random walk Metropolis algorithm. Thus, $\delta(n)$ is added if the acceptance rate calculated is greater than 0.44 and subtracted if the acceptance rate is less than 0.44.

$\delta(n)$ is set to be $\delta(n) = \min(0.01, 1/\sqrt{n})$ so the $\delta(n)$ approaches 0 as n grows large. This is to achieve one of two conditions to ensure the convergence of an adaptive MCMC

algorithm, which we will discuss further in this chapter. Also, ls_j is restricted in $[-M, M]$ for a really large constant M, so the algorithm can satisfy the other condition to guarantee the convergence. We will introduce two conditions that ensure the convergence of an adaptive algorithm in next section.

2.5 Convergence of Adaptive MCMC

Roberts and Rosenthal (2007) developed two conditions which ensures that an adaptive MCMC algorithm converges to its target distribution. They are Diminishing Adaption condition and Containment condition. Diminishing Adaption implies the adaption itself converges to zero for an adaptive MCMC algorithm. Containment implies the convergence times of the algorithm to the target distribution is bounded in probability.

Before we introduce more formal mathematical definitions for two concepts from Roberts and Rosenthal (2007), we introduce the notation for a general adaptive MCMC algorithm.

Let $\mathcal{X}$ be the state space with a corresponding Borel σ-algebra $\mathcal{F}$, and the target distributio π is defined on $\mathcal{X}$. Let $\mathcal{Y}$ be the index set of the collection of all the Markov chain kernels for the adaptive MCMC algorithm in consideration, and let γ be the element of $\mathcal{Y}$. Denote the each transition kernel for the adaptive MCMC algorithm $P_\gamma(x, \cdot)$, and we assume each P_γ leaves π stationary and is Harris ergodic. An adaptive MCMC algorithm at iteration n chooses γ by a $\mathcal{Y}$-valued random variable Γ_n, based on the information collected through $X_0, X_1, \ldots, X_n$ and/or auxiliary randomness. Thus, the transition probability for the adaptive MCMC algorithm for each $x \in \mathcal{X}$ and $A \in \mathcal{F}$,

$$\mathbf{P}[X_{n+1} \in A \mid X_n = x, \Gamma_n = \gamma, X_0, \ldots, X_{n-1}, \Gamma_0, \ldots, \Gamma_{n-1}] \;=\; P_\gamma(x, A)\,.$$

With this notation, the mathematical definition of Diminishing Adaption requires

$$\lim_{n\to\infty} \sup_{x\in\mathcal{X}} \|P_{\Gamma_{n+1}}(x,\cdot) - P_{\Gamma_n}(x,\cdot)\| \;=\; 0\,. \tag{2.3}$$

Let's define

$$M_\epsilon(x,\gamma) := \inf\{n \geq 1 : \|P_\gamma^n(x,\cdot) - \pi(\cdot)\| \leq \epsilon\}\,.$$

In words, $M_\epsilon(x,\gamma)$ is the time required to get to within ϵ of the stationary distribution π from the state x with a fixed transition kernal P_γ.

Containment requires, for all $\epsilon > 0$,

$$\{M_\epsilon(X_n,\Gamma_n)\}_{n=1}^\infty \text{ is bounded in probability}\,, \tag{2.4}$$

Theorem 4 (Roberts and Rosenthal (2007))**.** *Consider an adaptive MCMC algorithm defined on $(\mathcal{X},\mathcal{F})$. Assume Every transition kernel P_γ is Harris ergodic with the same stationary probability distribution π. (i.e. $\lim_{n\to\infty}\sup_{A\in\mathcal{F}}|P^n(x,A)-\pi(A)| = 0$ for all $x\in\mathcal{X}$). If the algorithm satisfies both Diminishing Adaption condition and Containment condition, then the algorithm converges to π as in*

$$\lim_{n\to\infty}\sup_{A\in\mathcal{F}}|\mathbf{P}(X_n\in A)-\pi(A)| = 0 \textit{ in probability}\,.$$

Chapter 3

Automatically Tuned General-Purpose MCMC via New Adaptive Diagnostics

3.1 Introduction

Markov Chain Monte Carlo (MCMC) is a technique widely used to sample from complex probability distributions, leading to numerous applications and methodological developments and theoretical advances (see e.g. Brooks et al. 2011). It is well-known that some Markov chains work much better than others in terms of convergence speed, asymptotic variance and/or mixing speed (see e.g. Rosenthal 2011), leading to questions of how to find more efficient chains.

Adaptive MCMC algorithms attempt to improve the Markov chain 'on the fly', using information from past iterations of the chain. This can significantly improve efficiency in practice (e.g. Haario et al. 2001, 2006; Roberts and Rosenthal 2009; Giordani and Kohn 2010; Vihola 2012; Turro et al. 2007). Unfortunately, most adaptive MCMC algorithms

are no longer Markovian, so convergence of the algorithm to the target distribution is much more difficult to establish and can sometimes fail (see e.g. Rosenthal 2004). This question has been investigated extensively, and researchers have proved the ergodicity of adaptive MCMC algorithms under various conditions (e.g. Haario et al. 2001, 2006; Giordani and Kohn 2010; Vihola 2012; Atchadé and Rosenthal. 2005; Andrieu and Moulines 2006; Andrieu and Atchadé 2007; Roberts and Rosenthal 2007; Fort et al. 2011). However, these results all require assumptions such as "Containment" or "simultaneous polynomial drift conditions" which are virtually impossible to verify directly in practical applications, and there aren't many widely applicable convergence results with easily-checkable assumptions. This means that when using adaptive MCMC in practice, there are usually no guarantees of even asymptotic convergence, and the user must simply "hope" that aberrant behaviour such as that exhibited in Rosenthal (2004) does not arise.

In this chapter, we present an algorithm which uses new adaptive diagnostics to determine when 'enough' adaption has already been done, i.e. when further adaption is not likely to lead to significant further improvements in efficiency. At this point, the adaption ceases, and the algorithm runs an ordinary non-adaptive MCMC algorithm for which convergence is guaranteed. In this way, our algorithm achieves the efficiency gains of adaptive MCMC, while avoiding the theoretical obstacles of typical adaptive MCMC algorithms which continue to adapt indefinitely. For definitiveness, we focus here on improving the proposal distribution for the Metropolis-Hastings algorithm (Metropolis et al. 1953; Hastings 1970), though similar ideas could also be applied in other adaptive MCMC contexts. We have developed a companion software package 'atmcmc' (Yang 2014), written in the R computer language (R Core Team 2014), to implement the algorithm introduced herein.

We note that various other software packages for adaptive MCMC are already available. One example is 'AMCMC' (Rosenthal 2007a,b) which employs an Adaptive Metropolis-within-Gibbs algorithm from Roberts and Rosenthal (2009). Another is the package 'Grapham'

(Vihola 2010a,b) for Adaptive Metropolis-within-Gibbs for graphical models with arbitrary block decompositions. Another example is 'adapMCMC' by Scheidegger (2012), which is based on the adaptive MCMC algorithm proposed by Vihola (2012) which tries to learn the shape of the target distribution while coercing the acceptance rate at the same time. A fourth example is 'FME' by Soetaert and Petzoldt (2014), based on Soetaert and Petzoldt (2010), whose function 'modMCMC' is an implementation of the Delayed Rejection Adaptive Metropolis (DRAM) method of Haario et al. (2006). These packages are all promising and useful, but they all involve infinite adaption, and thus require careful conditions to ensure convergence – in contrast to the algorithm herein which stops adapting once a 'good' proposal distribution is obtained and thus must converge by traditional Markov chain properties.

Section 2 of this chapter explains the idea behind the our algorithm. Section 3 provides some background information for our algorithm. Section 4 presents the details of the algorithm including all of the phases involved. Section 5 applies the algorithm to a number of MCMC examples. Section 6 provides some concluding comments.

3.2 Approach

It is well-known that a discrete-time Markov chain on a general state space converges eventually to its stationary distribution if it is ϕ-irreducible and aperiodic (e.g. Roberts and Rosenthal 2004). In practice, with time and resource constraints, we can't just rely on this eventual convergence and run a Markov chain to infinity. One concern is the time required to reach convergence.

Figure 3.1: Algorithm flowchart

Some Markov chains take unreasonably long time to reach convergence, especially in high dimension. We can improve the efficiency of a MCMC algorithm considerably by adapting. Since adaption destroys the Markov property of a Markov chain, the convergence of an adaptive MCMC algorithm has to be proven case-by-case. We want a more general algorithm so the user doesn't have to prove the convergence of an adaptive MCMC algorithm every time he tries different example, which sometimes is challenging. Thus, our goal here to make an algorithm which stops adapting once it obtains a 'good' proposal distribution, or, in other words, once the chain is tuned to improve the speed of the convergence.

Our focus in this chapter is to find a way to approximate the point where the adaption is not adding much value to the chain, thus allowing us to stop the adaptive algorithm. Also, to efficiently utilize the adaption which takes the past and current values of the chain to mimic the target distribution, we decide to take some pre-steps before the final adaption. The actual methods of adaption are not the main impotance here. You can change the adaption methods.

Since we are interested in a finite adaption here, we have to decide when to stop the adaption. To verify if the adaption is indeed improving the convergence speed of a Markov chain, we calculate the squared jumping distance, $(X_n - X_{n-1})^2$, for each iteration n. We want a Markov chain to explore the sample space of the target distribution quickly. In other words, we want a Markov chain to have a high mixing speed. The average squared jumping distances is one measure to show how well the chain is mixing by averaging the magnitudes of the movements from one state to the next. (This might not work as well with a multimodal target distribution as with a unimodal target distribution. We present an extension to our method for 'strongly multimodal targets' in Section 3.4.5.) If we see a general increase in the squared jumping distances as the adaptive algorithm runs, we presume the mixing of the chain is getting better. Since we have to account for random fluctuations in squared jumping distances, we calculate the average of squared jumping distances for a

fixed number of iterations and see if they are increasing. Roughly speaking, the more the chain moves for each iteration, the faster the chain will converge because it implies that the chain moves well throughout the entire state space, reducing the time to explore the full state space. Therefore, once the average squared jumping distance stops increasing, we assume further adaption would not significantly improve the mixing of the chain. We then stop the adaption, take the proposal distribution we get from the adaption, and run a standard MCMC algorithm for which all the properties and theories of a true Markov chain apply. We will use the Gelman-Rubin convergence diagnostic (Gelman and Rubin 1992; Brooks and Gelman 1998), which is explained briefly in Section 3.3, to check for the convergence of the chain when we run a standard MCMC algorithm.

However, in the collection of past values we use to modify the chain, we don't want to include too many values which have low probabilities of occurring in the target distribution. Thus, we want to discard a burn-in period until the chain reaches the mode of target distribution. We call this step to find the mode of the target distribution Transient phase. Only after then, we want to collect the values from the chain to fine-tune the chain.

At the same time we don't want our chain to take forever to reach the mode of the target distribution due to the bad proposal distribution for a specific target distribution. Thus, before we let the chain find the mode of the target distribution, or before Transient phase, we make the algorithm quickly and roughly adjust the chain using some adaptive method. We call this 1^{st} adaption phase.

Thus, our method consists of 1^{st} adaption phase, Transient phase, 2^{nd} adaption phase, and the sampling phase which we run the standard MCMC algorithm while checking for the convergence with Gelman-Rubin diagnostic. Figure 3.1 summarizes the key idea of our method. The technical details of our method are presented in Section 3.4.

3.3 Background

For Metropolis-Hastings algorithms, we can consider the acceptance rate, i.e. the rate at which a new proposed value at each iteration is accepted. We don't want the acceptance rate to be too high, which implies each proposed move is too small so the chain moves slowly throughout the whole state space. At the same time, we don't want the acceptance rate to be too low, either, since then the chain tends to get stuck at one value and hardly moves from it. In Roberts et al. (1997) it was shown that, for the symmetric random walk Metropolis algorithm with proposal distribution $N(0, \sigma^2 I_d)$, the optimal asymptotic acceptance rate is 0.234 as $d \to \infty$ if the target density function has the form $\pi(x) = \prod_{i=1}^{d} f(x_i)$ on $\mathbb{R}^d$. In Roberts and Rosenthal (2001), it was proved that the optimal acceptance rate is still 0.234 as $d \to \infty$ if the target density has the form of $\pi(x) = \prod_{i=1}^{d} C_i f(C_i x_i)$ where the $\{C_i\}$ are selected as i.i.d. positive random variables from some fixed probability distribution. Numerical studies also showed the optimal acceptance rate 0.234 is quite robust as it holds if d is as low as 5 (Gelman et al. 1996; Roberts and Rosenthal 2001) or the target density doesn't exactly have the form required to prove the theorem (Roberts and Rosenthal 2001). It was also found by numerical studies that the optimal asymptotic acceptance rate of one dimensional MCMC algorithm is 0.44 not 0.234 (Gelman et al. 1996; Roberts and Rosenthal 2001). Note that Gelman et al. (1996) and Roberts and Rosenthal (2001) demonstrated that it is not worth tuning for the *exact* optimal acceptance rate of 0.234, since if the average acceptance rate is between 0.15 and 0.5 then the Markov chain is at least 80% efficient compared to the optimal.

Another way to improve the convergence speed of a Markov chain is, for the symmetric random walk Metropolis algorithm with proposal distribution $N(0, \Sigma_p)$, to get the Σ_p proportional to the estimated covariance matrix of target distribution. This strategy was first suggested in Haario et al. (2001) and justified in Roberts and Rosenthal (2001). Intuitively,

this strategy seems promising since the more the proposal distribution is similar to the target distribution, the more likely the chain would propose a value that the target distribution would propose at each iteration given the current value X_n, and it wouldn't need as long a run of the Markov chain to converge to the target distribution. It is shown in Roberts et al. (1997) that if the target covariance matrix is multiplied by $2.38^2/d$ to obtain the covariance matrix Σ_p, then as $d \to \infty$, the proposal distribution $N(0, \Sigma_p)$ gives optimal convergence (again with acceptance rate 0.234) among all Gaussian proposal distributions.

Of course, in practice we don't know the target covariance matrix. However, we can use the empirical covariance matrix calculated from past chain values instead. That is, after some burn-in period, we use past values of the chain to estimate the covariance of the proposal distribution, and then multiply this empirically estimated matrix by $2.38^2/d$ and use that as our proposal covariance matrix (Haario et al. 2001; Roberts and Rosenthal 2009). As the chain runs for a long time, the collection of the past values from the chain gets closer to a sample from the true target distribution, and therefore the estimated covariance get closer to the true target covariance.

One last problem is that even when running a standard MCMC algorithm, we don't know when is a good time to stop the chain. In other words, we don't know when the chain convergence is achieved and we can take the values generated from the chain as a sample from the target distribution. Gelman and Rubin (1992) proposed a method to detect convergence of a Markov chain. They run several replicative Markov chains from an overdispersed starting distribution. They discard values from the first half of the chains and take the second half as a sample and diagnose convergence. They assume convergence is achieved if $\hat{R}_c := (\hat{V}/W)$(c.f.) is close to 1, where

$$\hat{V} \;=\; \frac{n-1}{n}\,W + \frac{B}{n} + \frac{B}{nm}\,,$$

W is the average sample variance from each replicative chain, and B/n is the variance of

sample mean from each chain. Here n is the sample size of each chain, m is the number of replicative chains, and the correction factor (c.f.) accounts for the difference in variances between the Student's t distribution and the normal distribution. If $\hat{R}_c$ is close to 1, this implies the variance of the sample mean from each chain is almost negligible, which hopefully indicates convergence to stationarity. The idea behind the Gelman-Rubin diagnostic is that if chains started from all over the state space each give us samples with similar distributions, then each chain must have reached the same distribution, the target. Therefore, starting the replicative chains from an 'overdispersed' starting distribution is crucial.

One pitfall of the Gelman-Rubin diagnostic $\hat{R}_c$ is that it implicitly assumes normality of the target distribution, in the sense that they only monitor means and variances to compare distributions of the replicative chains. To overcome this shortcoming, there is another paper by Brooks and Gelman (1998) extending the Gelman-Rubin diagnostic. They suggest the use of $(1-\alpha)*100\%$ confidence intervals, and define $\hat{R}_{interval}$ to be the ratio of the length of the total-sequence interval (found when all points from all replicative chains are thrown together as one sample) divided by the average length of the intervals from each of the replicative chains. They assume convergence is achieved if $\hat{R}_{interval}$ (instead of $\hat{R}_c$) is close to 1.

We next present the details of our algorithm, which combines all of the above ideas together to automatically tune MCMC, and hopefully achieve efficient convergence without sacrificing MCMC's theoretical guarantees.

3.4 Technical Details

The major breakdown of our algorithm is as follows. It consists of a number of distinct phases. We start with an Adaptive Metropolis-within-Gibbs algorithm (Roberts and Rosenthal 2009) to get a rough idea of scaling for each coordinate of the Markov chain ("1^{st} adaptive phase"). We continue this phase until we get an acceptance rate of every coordi-

nate in the neighbourhood of 0.44 (the optimal acceptance rate for one dimensional Markov chain). Note that we only need a very *rough* scaling estimate, so the neighbourhood range can be quite large. Then, we run a fixed (non-adaptive) Metropolis-within-Gibbs algorithm with this scaling ("transient phase"), and diagnose whether the chain has reached the *mode* of the target distribution. We do this by fitting a regression line to see if the chain values are trending, and continue until the regression signals that the chain becomes flat in every coordinate. Next, we employ an Adaptive Metropolis algorithm (Haario et al. 2001; Roberts and Rosenthal 2009) which adaptively updates the full-dimensional proposal covariance matrix Σ_p ("2^{nd} adaptive phase"). As mentioned earlier, the increase in the averaged squared jumping distance is used as a measure of the adaption improving the chain. We continue the Adaptive Metropolis algorithm until the average squared jumping distance stops increasing. At this point, we run a conventional Metropolis algorithm ("sampling phase"), and apply a Gelman-Rubin convergence diagnostic to divide the remaining run into two halves, one for burn-in and one for actual sampling from the target distribution.

We now describe the details of these various phases, one by one. The phases do involve various choices of approach and parameters, but they all appear to work well in practice and to be robust to different target distributions. Thus, the algorithm described below can be implemented directly (using our companion software package 'atmcmc' (Yang 2014)), without requiring any additional adjusting or tweaking by the user. And, as mentioned, since our algorithm uses only a finite amount of adaption followed by a conventional MCMC algorithm, asymptotic convergence to the target distribution is automatically guaranteed, without requiring the sorts of specialised arguments which are needed for most adaptive MCMC algorithms.

3.4.1 1^{st} Adaption Phase

To begin, we start with the Adaptive-Metropolis-within Gibbs algorithm introduced in Roberts and Rosenthal (2009) to get a rough idea of what is a 'good' scale for each coordinate of a Markov chain. We call this step 1^{st} adaption phase. Let X_0, X_1, X_2, ..., be a Markov chain process and Y be a new value proposed by a certain proposal distribution at each iteration. Given a current value of a Markov chain, X_n, Y is proposed by substituting $X_{n,j}$ with Y_j drawn by $Y_j \sim N(X_{n,j}, \sigma_j^2)$ where $X_{n,j}$ and Y_j are the j^{th} coordinate of X_n and Y, respectively, and σ_j^2 is the variance of the proposal distribution for the j^{th} coordinate. Then Y is either accepted ($X_{n+1} = Y$) or rejected ($X_{n+1} = X_n$) by the Metropolis rule. In short,

$$\begin{cases} X_{n+1} = Y \quad if \quad U < min(1, \pi(Y)/\pi(X_n)) \\ X_{n+1} = X_n \quad if \quad U \geq min(1, \pi(Y)/\pi(X_n)) \end{cases}$$

where $U \sim U(0,1)$ and $\pi(\cdot)$ is the target density function. This is done for every coordinate j, sequentially.

To begin, a single Markov chain is run from the randomly chosen initial point X_0. With adaption, for each coordinate j of a Markov chain, we try to achieve the acceptance rate of 0.44, which is known to be an approximately optimal acceptance rate for one dimensional Markov chain (Roberts and Rosenthal 2001). As in Roberts and Rosenthal (2009), we change the variance of the proposal distribution to alter the acceptance rate of a Markov chain since a proposal distribution with a larger variance tend to propose larger values, which would get rejected more often than values proposed by a proposal distribution with a smaller variance, and vice versa. Thus, for every 100 iteration, we calculate the acceptance rate of the past 100 iterations for each coordinate j, and we add $\epsilon = 0.05$ to $\log(\sigma_j)$ if the acceptance rate is higher than 0.44, and subtract ϵ from $\log(\sigma_j)$ if the acceptance rate is lower than 0.44. We

do this until the acceptance rate for every coordinate of the Markov chain falls between 0.28 and 0.60. If we get the acceptance rate for every coordinate in between 0.28 and 0.60, we run 100 more iterations with same σ_j's, which have made the acceptance rates to fall between 0.28 and 0.60, and monitor the acceptance rate for the past 200 iterations. If at least one acceptance rate from the coordinates falls outside of 0.28 and 0.60, then we adjust $\log(\sigma_j)$ for every 200 iterations until the acceptance rate for every coordinate comes between 0.28 and 0.60. Once we have the acceptance rate for every coordinate fall between 0.28 and 0.60, we run 200 more iterations with σ_j's unchanged and monitor the acceptance rate for past 400 iterations. If at least one acceptance rate falls outside of 0.28 and 0.60, we adjust $\log(\sigma_j)$ for every 400 iterations until the acceptance rate for every coordinate fall between 0.28 and 0.60. If the acceptance rate for every coordinate from past 400 iterations falls between 0.28 and 0.60, we stop the chain and save σ_j for every coordinate.

Note that here the acceptance rate is a continuous function of proposal variance. If we want to increase (decrease) the acceptance rate by a bit, we have to decrease (increase) the proposal variance by a bit accordingly. We can always get the acceptance rate to fall into some desired range (easier if the range is big) as long as the target density and the proposal density is positive everywhere in the state space and the shift in the proposal variance is not too big at each adjustment.

3.4.2 Transient Phase

Next, we try to find if there is any transient phase for the Markov chain since we don't want to start final adaption when the values generated, which will be used for the adaption, are far from where the major mass of the target distribution is. We want a burn-in phase to discard the part of the chain which mostly consists of values in the low probability zones under the target distribution. We call this transient phase.

We employ a standard Metropolis-within-Gibbs algorithm with proposals for each coordinate drawn from $Y_j \sim N(X_n, \sigma_j^2)$ with σ_j^2 determined by the 1^{st} adaption phase. To check if the chain is moving towards the mode of target distribution, for every 200 iterations, the values generated for each coordinate j of X are averaged, and with averages of 5 consecutive 'batches' for each coordinate j, a linear model is fitted to see if there is any trend in the j^{th} coordinate of the chain. The specific values 200 and 5 are somewhat arbitrary choices, but we have found that they work well when we run the algorithm. The user has flexibility to pick some other numbers, as long as he/she makes sure the algorithm has enough points (for example, 5) to run a regression but also at the same time two numbers (for example, 200 and 5) are not too big so it doesn't take too long to test whether the chain is trending or not.

We use a regression method to make sure the chain values are moving to only one direction, neither increasing nor deceasing. If a regression method confirms that the chain values show a linear trend, we presume that the chain is still moving to a local mode. The p-value for the slope coefficient is used to determine whether there is any linear trend. If p-value for every coordinate is greater than 0.1, the chain gets stopped and this phase ends. We have found that a p-value of 0.1 is a reasonable cutoff for our purpose. The p-value below 0.1 is the point where people start to talk about any sign of significance in statistics, although many prefer a lot lower p-value than 0.1 to actually claim the significance in practice. Here, we are detecting non-significance, so 0.1 seems to be a convenient threshold which appears to be robust in practice. Plus, if for some reason, the p-value cutoff misses a trending chain (which we believe has a very low chance of happening), we still have the last phase which will run a standard MCMC algorithm. Therefore, at a minimum, the algorithm will converge to the target distribution even though we might lose some efficiency.

3.4.3 2^{nd} Adaption Phase

Here, we slightly modify the Adaptive Metropolis algorithm introduced in Haario et al. (2001) and Roberts and Rosenthal (2009)) to find the proposal distribution that has a similar covariance structure with the target distribution. This phase is called 2^{nd} adaption phase. The proposal, Y, is drawn from $Y \sim N(X_n, c\Sigma_{n+1})$, and again the accept/reject is by the Metropolis rule. Σ_n is found by calculating the covariance matrix of all past values generated by the chain from the point the trending stops in the transient phase to X_{n-1}, and $c = 2.38^2/d$. After 200 iterations in this phase, if the acceptance rate is too low (less than 0.02), then we reduce c by a factor of d, to $2.38^2/d^2$, to make the scale of the proposal distribution smaller thus increasing the acceptance rate, and start this phase again with the last value from the transient phase as the starting value. We want to stop the adaption when further adaption does not improve the chain. To check whether the adaption is improving the chain or not, for every 200 iterations, we calculate the average squared jumping distance for each coordinate j, and again with averages of 5 consecutive 'batches' for each coordinate j, we fit a linear model to see if there is any trend in squared jumping distance for each coordinate. If average squared jumping distance stops increasing, we presume the mixing of the chain stops improving, and we stop this phase. Thus, we stop this phase when, for every coordinate, the p-value for the slope coefficient of the regression is greater than 0.1.

3.4.4 Sampling Phase

Finally, we apply the symmetric random walk Metropolis algorithm with proposal Y drawn from $Y \sim N(X_n, \Sigma_p)$. Σ_p is the last Σ_n obtained from the 2^{nd} adaption phase multiplied by the final value of constant c. We use both the Gelman-Rubin diagnostic $\hat{R}_c$ and the extension $\hat{R}_{interval}$ to monitor convergence of the Markov chain, since $\hat{R}_{interval}$ doesn't require the normality assumption that $\hat{R}_c$ does. To apply these diagnostics, we run $k = 10$ replicative

chains simultaneously. Besides the last value of the 2^{nd} adaption phase used as the initial point of one replicative chain, the initial points for each of $k-1$ replicative chains are drawn from $U(d_j-(e_j-d_j)/4, e_j+(e_j-d_j)/4)$ for each coordinate where d_j and e_j are the minimum and maximum value, respectively, for each coordinate j of the chain found from the point the trending stops in the transient phase to the end of the 2^{nd} adaption phase. After some burn-in period, we discard the values from the first half of this phase, and calculate $\hat{R}$s with the values from the second half of the phase. When both $\hat{R}$s for every coordinate are close to 1, we stop the chain and take the values from the second half of this phase as our sample from the target distribution. For our runs of the algorithm, including runs on the examples described in Section 3.5, we stopped the algorithm if $\hat{R}_c$ and $\hat{R}_{interval}$ falls in between 0.9 and 1.1. The user of our algorithm can make the cutoff values more strict if he/she desires. We call this phase sampling phase. Note that we throw values from all replicative chains together in our sample.

Note that most of numerical values used in the algorithm can be changed by the user in the R package 'atmcmc'. The choice of these numerical values does affect the performance of MCMC, and the numbers specified in this section are the default values built in the R package. For details, see the reference manual of the R package 'atmcmc' (Yang 2014).

3.4.5 Extension for Strongly Multimodal Targets

Suppose a target distribution has multiple local modes, and there is at least one mode that is strongly separated from any of the other modes. In other words, the region between this mode and any other modes is at low density. Since accepting a proposal value that is in a much lower density region compared to the current state is a low probability event, unless a direct proposal of a point in another mode occurs from time to time, there is a chance a Markov chain gets stuck in that mode. We will call a target distribution like this 'strongly

multimodal'.

With a 'strongly multimodal' target distribution, we run multiple Markov chains with different initial points drawn randomly from $U(a, b)$, $a, b \in \mathbb{R}^d$, for the 1^{st} adaption phase and transient phase as the chains find different modes. It is important to run enough chains with widespread initial points to find all modes in the target distribution. (Note that we do *not* assume that we know the location of the modes in advance, nor even the number of modes.)

Remark. *'If' we know the number of modes in advance but do not know the location of the modes, here is a rough guideline for the number of chains to be used. Suppose we have r different modes, and we plan to run k different chains. Let's say we want less than or equal to 5% chance of missing at least one mode. We can use a formula $k = \frac{log(0.05/r)}{log((r-1)/r)}$ for the number of chains to be used, since the probability of missing at least one mode purely by chance is bounded above by $r(\frac{r-1}{r})^k$. Note that this is only a rough guideline, and it is up to a user's judgment to decide how many chains to run while considering both the computation cost and the probability of missing mode(s).*

Once all these different chains find different modes, we calculate the mean and standard deviation of each Markov chain from the segment off the transient phase where the chain is not trending, and we use these means and standard deviations to determine if the chains found different modes. For any two chains, for at least one coordinate, if the difference in two chain means is greater than at least one of two chain standard deviations, we consider the two chains to have reached different modes. With this selection process, we only collect the chains with different modes.

Then we go into the 2^{nd} adaption phase with the chains we are left with after the selection process. Once each of these chains stops in the 2^{nd} adaption phase, we get different Σ_i for each chain. Again, with the values generated from the 2^{nd} adaption phase, we check if each

Markov chain still stays at different modes from each other.

After this, we start with some initial point, and run a MCMC algorithm that we are about to describe. The initial points are the last point of the 2^{nd} adaption phase from each chain and some randomly drawn points from $\sum_{i=1}^{r} U(a_i, b_i)/r$, where i indexes for each of r chains with different modes and (a_i, b_i) are decided based on the values generated from the 2^{nd} adaption phase. We don't use values from any part of the transient phase here since there is a chance that two or more chains with different modes merge during the 2^{nd} adaption phase, giving us a different number of 'unique' chains after the 2^{nd} adaption phase than right after transient phase. Keep in mind that we want the overdispersed starting distribution for the Gelman-Rubin diagnostic. As the Markov chain runs, we need to evaluate which mode the current value $X_n = x$ is the closest to. The rule for determining this is

$$\text{mode}(x) = \underset{i}{\text{argmin}}\ D_i$$

where $D_i = \max_j d_{ij}$ with $d_{ij} = |x_j - m_{ij}|/\sigma_{ij}$. Here i indexes the multiple chains with different modes, and j indexes for the coordinate of X. Also m_{ij} and σ_{ij} are the mean and standard deviation of chain i, $i \in \{1, ..., r\}$, and coordinate j, $j \in \{1, ..., d\}$, calculated from the values generated from 2^{nd} adaption phase. Again, we don't include values from any part of the transient phase when calculating the mean and standard deviation of each chain i.

Suppose $\text{mode}(X_n) = k$. Then we update the Markov chain in the sampling phase as follows.

1. With probability $1 - \alpha$, we propose Y by

$$Y|X_n \sim N(X_n, c\Sigma_k)$$

where Σ_k is the variance obtained from the 2^{nd} adaption phase for the chain k. We shall

write $q_{1k}(x, y)$ for the probability of proposing y given x using this rule. That is,

$$q_{1k}(x,y) = \begin{cases} \frac{1}{\sqrt{(2\pi)^s |c\boldsymbol{\Sigma}_k|}} \exp\left(-\frac{1}{2c}(y-x)^T {\boldsymbol{\Sigma}_k}^{-1}(y-x)\right), & \text{if mode}(x) = \text{mode}(y) = k \\ 0, & \text{otherwise} \end{cases}$$

Note that here $q_{1k}(x, y) = q_{1k}(y, x)$, since q_{1k} depends only on $|y - x|$. y is rejected if mode$(x) \neq$ mode(y). Otherwise, y is accepted or rejected based on the Metropolis rule.

2. With probability α, we propose Y by proposing each coordinate of Y as

$$Y_j | X_{n,j} = \frac{\sigma_{lj}}{\sigma_{kj}}(X_{n,j} - m_{kj}) + m_{lj} \tag{3.1}$$

Here l is a random draw from all different chains i excluding chain k, where the probability of drawing some chain l is uniform. Thus, σ_{lj} is the univariate standard deviation of the chain l and the coordinate j, from the sample obtained for the $2^n d$ adaption phase, and σ_{kj} is the univariate standard deviation of the chain k and the coordinate j. In other words, if there are r different chains with all different modes,

$$q_2(x,y) = \begin{cases} \frac{1}{r-1}, & \text{if } y_j = \frac{\sigma_{lj}}{\sigma_{kj}}(x_j - m_{kj}) + m_{lj},\ j = 1, ..., s,\ l = 1, ..., r,\ l \neq k \\ 0, & \text{otherwise} \end{cases}$$

where $q_2(x, y)$ is the probability of suggesting y given x using the rule (3.1) . Note that for all (x, y), $q_2(x, y) = q_2(y, x)$. y is rejected if mode$(x) =$ mode(y). Otherwise, y is accepted or rejected based on the Metropolis rule.

Thus, if $P(x,\cdot)$ is the transition probability for X and $\text{mode}(x) = \text{mode}(y) = k$, then

$$\begin{aligned}
\pi(dx)P(x,dy) &= [s^{-1}\pi_u(x)dx][q(x,y)a(x,y)dy] \\
&= [s^{-1}\pi_u(x)][(1-\alpha)q_{1k}(x,y)][\min(1, s^{-1}\pi_u(y)/s^{-1}\pi_u(x))]dxdy \\
&= s^{-1}[(1-\alpha)q_{1k}(x,y)][\min(\pi_u(x),\pi_u(y))]dxdy \\
&= s^{-1}[(1-\alpha)q_{1k}(y,x)][\min(\pi_u(y),\pi_u(x))]dydx \\
&= [s^{-1}\pi_u(y)dy][q(y,x)a(y,x)dx] \\
&= \pi(dy)P(y,dx)
\end{aligned}$$

for some normalizing constant s for $\pi_u(\cdot)$, an unnormalized density function of π. $q(x,y)$ is the probability of suggesting y given x for the Markov chain of interest, and $a(x,y)$ is the probability of accepting y given x.

If $\text{mode}(x) \neq \text{mode}(y)$, then

$$\begin{aligned}
\pi(dx)P(x,dy) &= [s^{-1}\pi_u(x)dx][q(x,y)a(x,y)\sum_{i=1,\, z_i \neq x}^{r} \delta_{z_i}dy] \\
&= [s^{-1}\pi_u(x)][\alpha q_2(x,y)\sum_{i=1,\, z_i \neq x}^{r} \delta_{z_i}][\min(1, s^{-1}\pi_u(x)/s^{-1}\pi_u(y))]dxdy \\
&= s^{-1}[\frac{\alpha}{r-1}\sum_{i=1,\, z_i \neq x}^{r} \delta_{z_i}][\min(\pi_u(x),\pi_u(y))]dxdy \\
&= s^{-1}[\frac{\alpha}{r-1}\sum_{i=1,\, z_i \neq y}^{r} \delta_{z_i}][\min(\pi_u(y),\pi_u(x))]dydx \\
&= [s^{-1}\pi_u(y)dy][q(y,x)a(y,x)\sum_{i=1,\, z_i \neq y}^{r} \delta_{z_i}dx] \\
&= \pi(dy)P(y,dx)\,.
\end{aligned}$$

where $\delta_{z_i}(\cdot)$ is a point-mass at $z_i \in \mathcal{X}$. Note that $r-1$ z_i's are determined by the rule (3.1)

given x and the last z_i is x itself.

Additionally, recall that when $\pi_u(x) = 0$, we always accept the new proposal y. If $\pi_u(x) = 0$ and $\pi_u(y) \neq 0$, then $\pi(dx)P(x, dy) = \pi(dy)P(y, dx) = 0$ since $P(y, dx) = 0$ and vice versa. If $\pi_u(x) = 0$ and $\pi_u(y) = 0$, then it is trivial.

To sum up, the Markov chain we construct above is reversible with respect to $\pi(\cdot)$.

In our runs of the proposed scheme, we used $\alpha = 0.05$. α cannot be too big since it is not desired that the mode-to-mode jump happen too frequently, and it cannot be too small resulting in mode-to mode-jump happening only few times, which reduces the effectiveness of our algorithm for the 'strongly multimodal' targets. The user of our scheme (and the user of the R package 'atmcmc') has some freedom to choose α as long as it is not too big or too small due to the reasons just described.

Convergence of the Markov chain is still diagnosed the same way explained in the main algorithm, and, again, the final sample obtained is the collection of all values from the second half of all replicative chains created for Gelman-Rubin diagnostics. We assume throwing all points from all replicative chains together will not distort the end result as we believe the convergence test by Gelman-Rubin can be only passed when the mixing of each replicative Markov chain is 'good'.

3.5 Applications

In this section, we present a number of different applications of our algorithm. The plots of the sampling phases come from one replicative chain out of 10, which uses the last value of the 2^{nd} adaption phase as the starting point for the sampling phase. For ease of identification, each phase is coloured differently in the trace plots: purple for the 1^{st} adaption phase, orange for the transient phase, red for the 2^{nd} adaption phase, and blue and green for the first and second halves of the sampling phase. Thus, the green segment in each trace plot is what we

take as our actual sample, and is also what we use to calculate the displayed acceptance rates in the tables. Also note that in the trace plots, the iteration number for the 1^{st} adaption and transient phases is counted coordinate-by-coordinate as the trace plots are presented coordinate-by-coordinate. More clearly, the first update of coordinate 1 is labeled iteration 1 in the trace plot for coordinate 1, and the first update of the coordinate 2 (second update for the whole algorithm) is labeled iteration 1 in the trace plot for the coordinate 2 and so on.

In each example, two tables are given. The first one displays the 10 estimates of parameters obtained from 10 independent runs of the full algorithm in Section 3.4. The second one displays the summary statistics of these 10 estimates: mean, standard deviation, minimum and maximum. The second tables also contain the true values of the parameters being estimated or estimates of the parameters using some other method, so the reader can compare and see how the algorithm presented in this chapter has performed.

Note that all the starting points for the 1^{st} adaption phases are arbitrary chosen as $0.1*\mathbf{1}$, except in the 'strongly multimodal' case. For the examples presented in Section 3.5.1, 3.5.2, 3.5.3, and 3.5.4, we did not use the scheme for 'strongly multimodal targets' from Section 3.4.5. The scheme from Section 3.4.5 was applied only for the example in Section 3.5.5.

3.5.1 Multivariate Normal Distribution

First, we take a 9-dimensional multivariate normal distribution as our target distribution. Each component of target mean, μ, was randomly drawn from $N(0, 1000^2)$ and target variance were constructed by $\Sigma\Sigma^t$ where each component of Σ was drawn from $N(0, 20^2)$. The target distribution of interest is $N(\mu, \Sigma\Sigma^t)$. The results of 10 different runs are shown in Table 3.1 and Table 3.2.

Table 3.1: Results of MCMC: Multivariate Normal. Results of 10 independent runs of full algorithm in Section 3.4.

	Estimates									
	106.41	104.39	100.40	103.95	103.68	100.95	105.98	102.33	99.95	102.40
	-525.14	-524.19	-524.41	-524.70	-526.58	-525.22	-525.52	-524.31	-522.74	-523.95
	-863.71	-863.87	-859.42	-856.61	-862.88	-862.62	-866.63	-859.77	-868.21	-866.57
	407.99	403.73	403.35	404.56	406.61	409.45	409.35	406.99	406.59	404.20
μ	976.56	971.01	975.57	967.18	975.00	976.11	977.07	974.82	979.93	975.85
	-451.47	-448.95	-449.23	-445.73	-449.28	-446.40	-450.20	-445.58	-446.73	-451.32
	641.23	640.33	637.31	636.82	643.90	644.32	646.91	641.47	648.55	644.17
	-556.63	-557.20	-559.81	-564.46	-560.43	-561.50	-561.68	-562.48	-562.97	-556.53
	796.44	797.07	794.05	797.12	795.07	787.79	798.65	792.79	791.75	796.50
	Acceptance Rates									
	0.2685	0.2565	0.2616	0.2771	0.2469	0.2750	0.2765	0.2755	0.2775	0.3031
	Runtime (in seconds)									
	30.28	29.54	28.90	28.95	33.64	29.59	30.08	29.55	31.54	31.28

Table 3.2: Summary Statistics for the Estimates in Table 3.1: Multivariate Normal. μ is the true mean of the target distribution.

Mean of Estimates	SD of Estimates	Min Estimate	Max Estimate	μ
103.04	2.23	99.95	106.41	103.54
-524.68	1.03	-526.58	-522.74	-524.46
-863.03	3.63	-868.21	-856.61	-862.79
406.28	2.25	403.35	409.45	405.96
974.91	3.50	967.18	979.93	974.04
-448.49	2.23	-451.47	-445.58	-448.01
642.50	3.81	636.82	648.55	642.51
-560.37	2.79	-564.46	-556.53	-561.15
794.72	3.23	787.79	798.65	796.02

As we see from Table 3.2, the estimates from the runs look to be close enough with the true mean μ. Trace plots for the first run can be found in Figure 4.2. The mixing of Markov Chain for the sampling phase, the blue and green phase in the figure, look to be good.

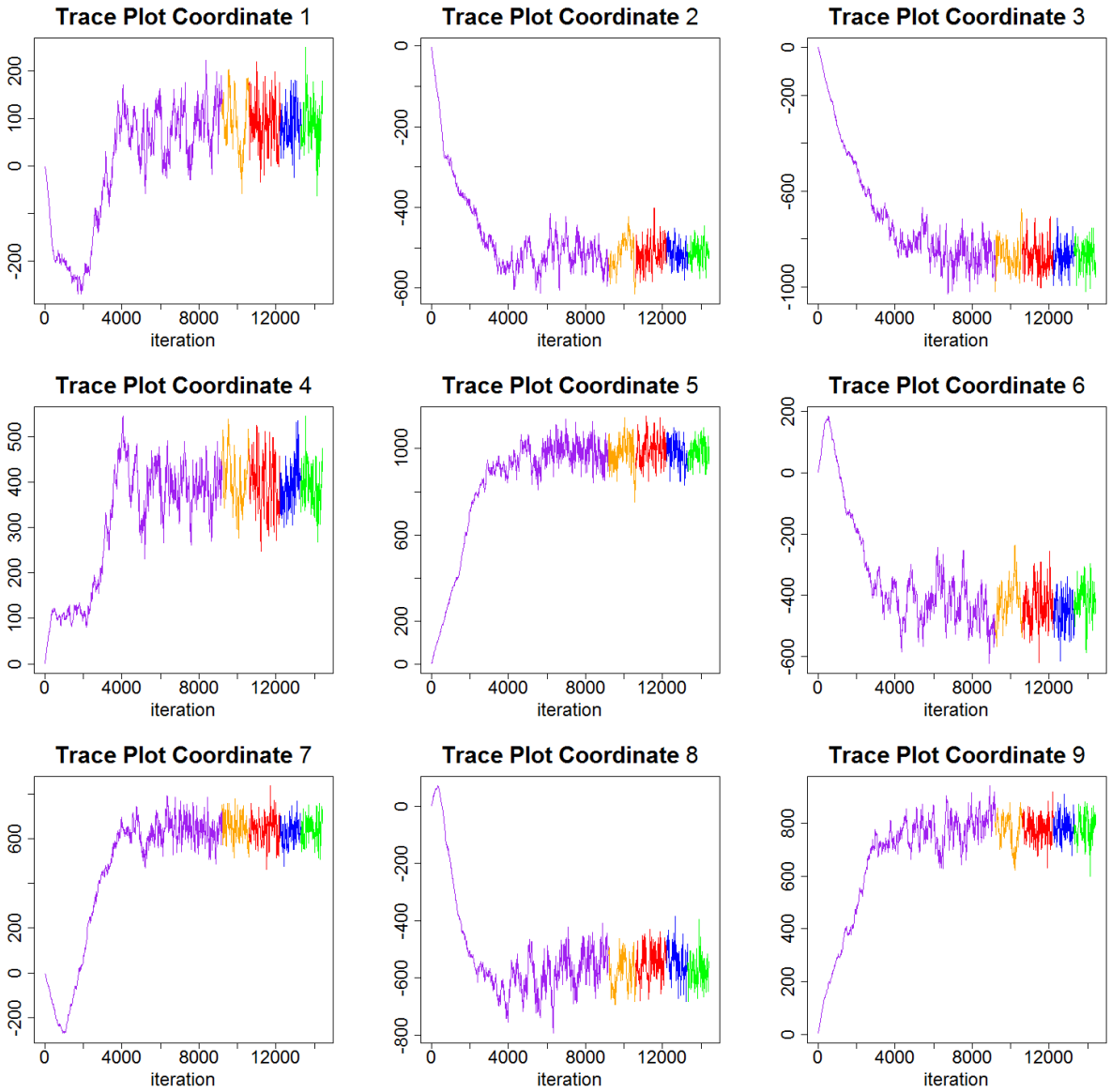


Figure 3.2: Trace Plots for Multivariate Normal

3.5.2 Logistic Regression

Next, we run a MCMC with a simple logistic regression model. The data used here is from data(logit) in the R package 'mcmc' (Geyer and Johnson 2014). It is a simulated logistics regression dataset, with a hundred data points and five variables. We name the five

variables y, x_1, x_2, x_3 and x_4. y is a Bernoulli response and x_1, x_2, x_3 and x_4 are quantitative predictors. We know the basic structure of a logistic regression is

$$\begin{cases} f(y_1, ..., y_n|\beta, X) = \prod_{i=1}^{n} P(y_i = 1|\beta, X)^{y_i} P(y_i = 0|\beta, X)^{1-y_i} \\ P(y_i = 1|\beta, X) = 1/(1 + exp(-\eta)) \\ \eta = ln[P(y_i = 1|\beta, X)/P(y_i = 0|\beta, X)] = \beta X. \end{cases}$$

If we put a prior $\beta = (\beta_0, \beta_1, ..., \beta_k)^T \sim N(0, 4)$, the posterior distribution of interest is

$$f(\beta|y_1, ..., y_n, X) \propto exp(-\sum \beta_j^2/8)[(\frac{1}{1 + exp(-\beta X)})^{\sum y_i}][(1 - \frac{1}{1 + exp(-\beta X)})^{n-\sum y_i}].$$

Again, the results of 10 different runs are shown in Table 3.3 and Table 3.4, and the trace plots for the first run are shown in Figure 3.3. We see our algorithm worked fine as we compare its results with MCMC estimates via R package 'mcmc' and with GLM estimates. Trace plots show a good mixing of the Markov chain.

Table 3.3: Results of MCMC:Logistic Regression. Results of 10 independent runs of full algorithm in Section 3.4.

	Estimates									
β_0	0.6618	0.6469	0.6671	0.6498	0.6486	0.6494	0.6678	0.6566	0.6654	0.6541
β_1	0.7975	0.7878	0.8240	0.7982	0.8129	0.8039	0.8216	0.8089	0.8082	0.8191
β_2	1.1848	1.2173	1.1621	1.1729	1.1761	1.1593	1.1635	1.1804	1.1574	1.1583
β_3	0.5192	0.5039	0.5120	0.5127	0.5016	0.5063	0.4967	0.5114	0.5077	0.4873
β_4	0.7095	0.7188	0.6875	0.7174	0.7148	0.7192	0.7347	0.7093	0.7229	0.7180
	Acceptance Rates									
	0.2728	0.2925	0.2927	0.2901	0.3028	0.2863	0.2876	0.2901	0.2786	0.2891
	Runtime (in seconds)									
	6.22	6.47	6.18	6.06	6.24	6.16	6.21	6.18	6.33	6.05

Table 3.4: Summary Statistics for the Estimates in Table 3.3: Logistic Regression. Results of a run from R package 'mcmc' and GLM estimates are also presented, for comparison.

	Mean of Estimates	SD of Estimates	Min Estimate	Max Estimate	Rpackage mcmc	GLM
β_0	0.6567	0.0082	0.6469	0.6678	0.6634	0.6328
β_1	0.8082	0.0117	0.7878	0.8240	0.7629	0.7390
β_2	1.1732	0.0183	1.1574	1.2173	1.2074	1.1137
β_3	0.5059	0.0091	0.4873	0.5192	0.5315	0.4781
β_4	0.7152	0.0121	0.6875	0.7347	0.7408	0.6944

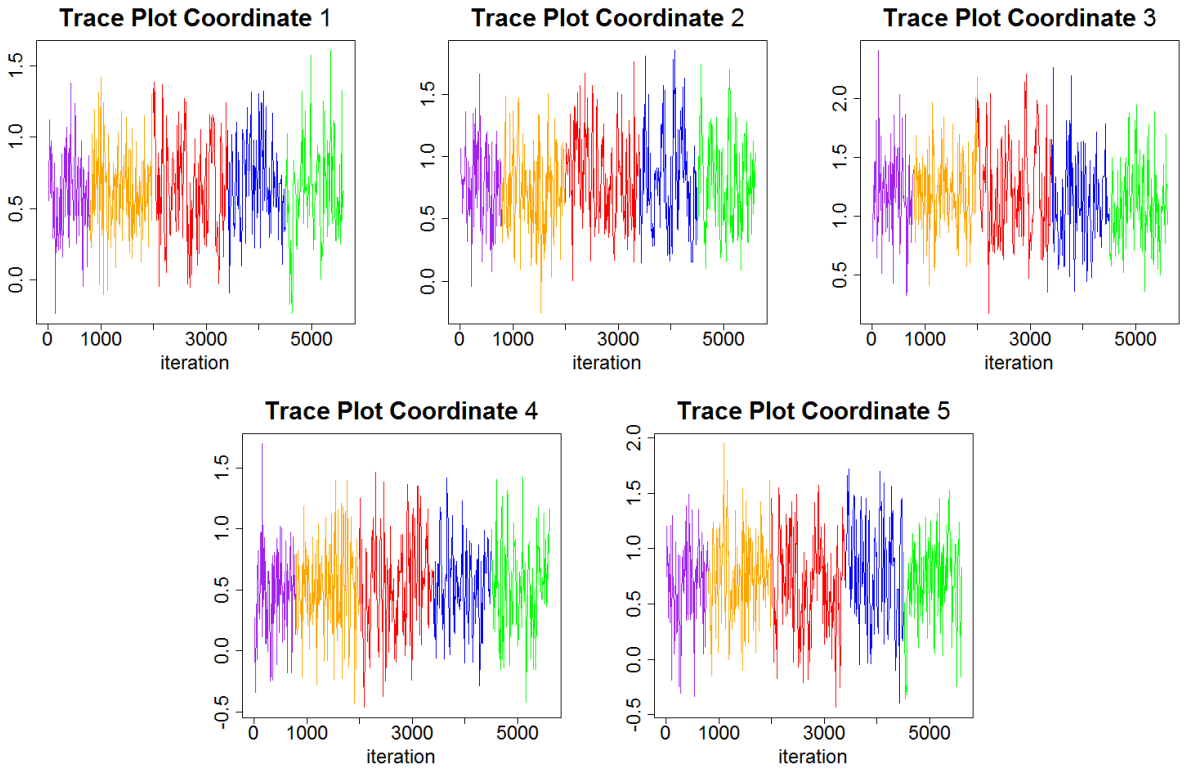


Figure 3.3: Trace Plots for Logistic Regression. Each coordinate corresponds to the parameters listed in Table 3.3 as ordered.

3.5.3 Pump Failure Data

This is the data from Gaver and O'Muircheartaigh (Gaver and O'Muircheartaigh 1987). It is shown in Table 4.1

Table 3.5: Pump Failure Data

Obs. no.	1	2	3	4	5	6	7	8	9	10
y_i	5	1	5	14	3	19	1	1	4	22
t_i	94.320	15.720	62.880	125.760	5.240	31.440	1.048	1.048	2.096	10.480

We followed the Bayesian set-up from George et al. (1993) to construct the posterior

distribution.

$$f(y_1, ..., y_n|\lambda_1, ..., \lambda_n) = \prod_{i=1}^{n} Poisson(\lambda_i t_i)$$

where n=10, $\lambda_i \sim G(\alpha, \beta)$, $\alpha \sim exp(1)$ and $\beta \sim G(0.1, 1)$. Thus, the posterior distribution of the parameters is

$$f(\lambda_1, ..., \lambda_n, \alpha, \beta|y_1, ..., y_n) \propto e^{-\alpha}\beta^{0.1-1}e^{-\beta}\prod_{i=1}^{n}\frac{\beta^{\alpha}}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}(\lambda_i t_i)^{y_i}e^{-\lambda_i t_i}$$

The results are shown in Table 3.6, Table 3.7 and Figure 4.3. We bring the results from OpenBUGS website (Lunn et al. 2009) to compare with our estimates.

Table 3.6: Results of MCMC: Pump Failure Data. Results of 10 independent runs of full algorithm in Section 3.4.

	Estimates									
λ_1	0.0579	0.0626	0.0598	0.0591	0.0603	0.0593	0.0589	0.0595	0.0574	0.0598
λ_2	0.1023	0.1042	0.1014	0.1077	0.1011	0.1042	0.0938	0.1033	0.0965	0.1065
λ_3	0.0889	0.0911	0.0926	0.0923	0.0873	0.0869	0.0903	0.0856	0.0900	0.0870
λ_4	0.1159	0.1163	0.1165	0.1160	0.1126	0.1176	0.1173	0.1150	0.1153	0.1187
λ_5	0.5629	0.6008	0.5884	0.5700	0.5936	0.5954	0.5947	0.6074	0.6090	0.5966
λ_6	0.6030	0.6071	0.5956	0.6140	0.6085	0.5955	0.5941	0.6101	0.6122	0.5973
λ_7	0.9148	0.8865	0.8217	0.8945	0.8879	0.8324	0.8332	0.8539	0.8697	0.8685
λ_8	0.9548	0.8217	0.8764	0.8093	0.8847	0.8935	0.9589	0.9104	0.9402	0.9671
λ_9	1.5930	1.5932	1.5768	1.6202	1.5692	1.4670	1.5332	1.5542	1.6088	1.6102
λ_{10}	2.0065	1.9909	1.9897	2.0038	1.9872	2.0169	1.9948	1.9624	1.9961	2.0507
α	0.6963	0.7026	0.7009	0.7123	0.6864	0.6878	0.7039	0.6965	0.6996	0.6817
β	0.9183	0.9355	0.9080	0.9497	0.9152	0.9498	0.9453	0.8963	0.9304	0.9180
	Acceptance Rates									
	0.1690	0.1837	0.1660	0.1612	0.1694	0.1533	0.1558	0.1593	0.1740	0.1978
	Runtime (in seconds)									
	31.79	24.53	24.23	23.03	25.06	23.64	25.22	22.69	25.95	30.55

Table 3.7: Summary Statistics for the Estimates in Table 3.6: Pump Failure Data. Results from the OpenBUGS website, obtained via Gibbs sampler, are also presented, for comparison.

	Mean of Estimates	SD of Estimates	Min Estimate	Max Estimate	BUGS
λ_1	0.0595	0.0014	0.0574	0.0626	0.05986
λ_2	0.1021	0.0042	0.0938	0.1077	0.1015
λ_3	0.0892	0.0024	0.0856	0.0926	0.08899
λ_4	0.1161	0.0017	0.1126	0.1187	0.1156
λ_5	0.5919	0.0149	0.5629	0.6090	0.6043
λ_6	0.6037	0.0076	0.5941	0.6140	0.6121
λ_7	0.8663	0.0306	0.8217	0.9148	0.899
λ_8	0.9017	0.0557	0.8093	0.9671	0.9095
λ_9	1.5726	0.0458	1.4670	1.6202	1.587
λ_{10}	1.9999	0.0229	1.9624	2.0507	1.995
α	0.6968	0.0092	0.6817	0.7123	0.6867
β	0.9267	0.0184	0.8963	0.9498	0.9024

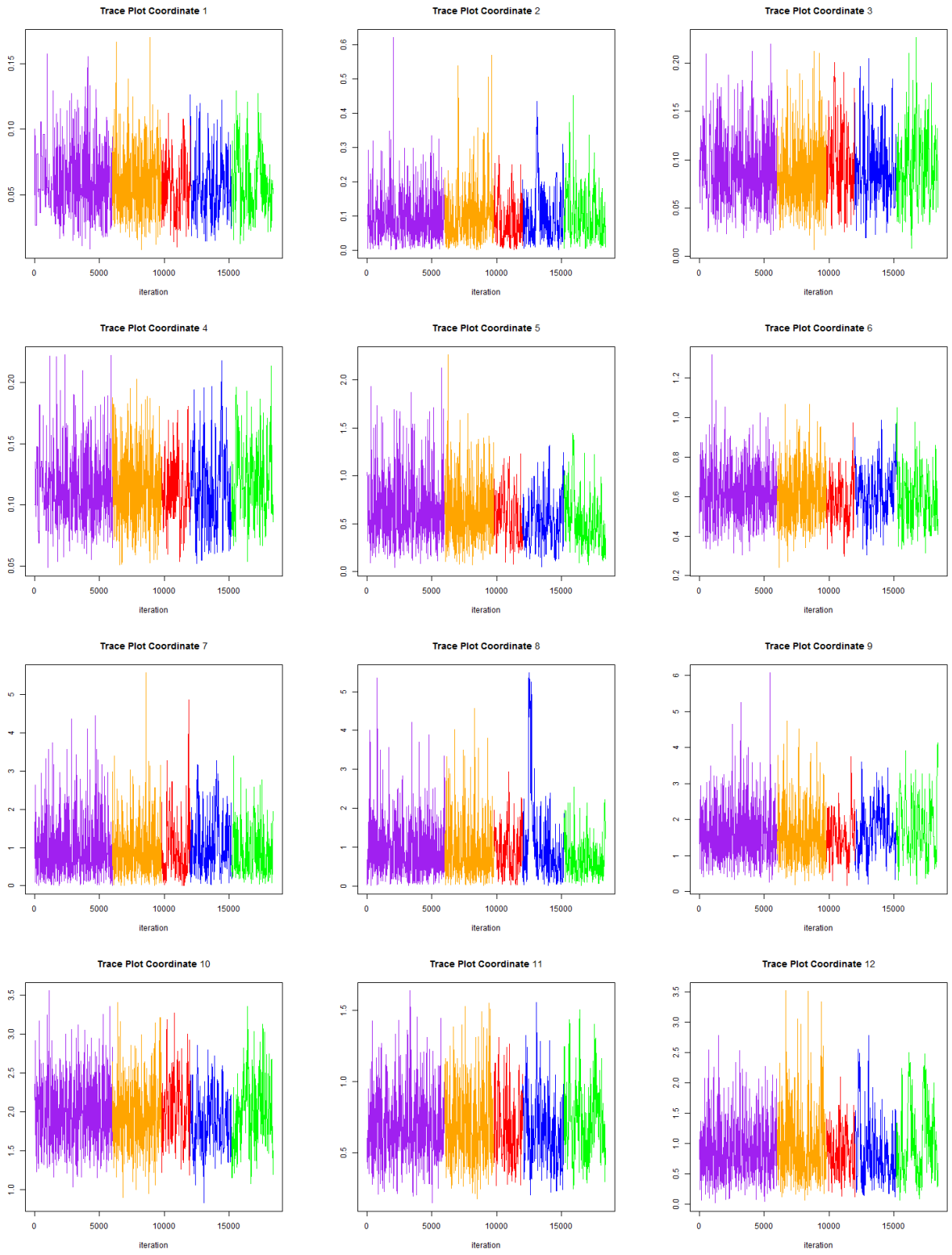


Figure 3.4: Trace Plots for Pump Failure Data. Each coordinate corresponds to the parameters listed in Table 3.6 as ordered.

3.5.4 Variance Components Model (VCM)

The Variance Components Model (VCM) is a well-known example in Bayesian statistics. The structure of the model can be found in Roberts and Rosenthal (2004) and Gelfand and Smith (1990). In short, the model is constructed as:

$$y_{ij}|\theta_i, \sigma_e^2 \sim N(\theta_i, \sigma_e^2), \qquad i = 1, 2, ..., K, j = 1, 2, ..., J$$

where $\theta_i|\mu, \sigma_\theta^2 \sim N(\mu, \sigma_\theta^2)$. $\theta_i|\mu, \sigma_\theta^2$ are independent of each other. The distributions of hyperparameters are: $\sigma_\theta^2 \sim IG(a_1, b_1)$, $\sigma_e^2 \sim IG(a_2, b_2)$ and $\mu \sim N(\mu_0, \sigma_0^2)$. Thus, the full posterior for the VCM is

$$f(\sigma_\theta^2, \sigma_e^2, \mu, \theta_i|y_{ij}) \propto (\sigma_\theta^2)^{-(a_1+1)} e^{-b_1/\sigma_\theta^2} (\sigma_e^2)^{-(a_2+1)} e^{-b_2/\sigma_e^2} e^{-(\mu-\mu_0)^2/2\sigma_0^2} \times \prod_{i=1}^{K} \frac{e^{\theta_i-\mu)^2/2\sigma_\theta^2}}{\sigma_\theta} \prod_{i=1}^{K}\prod_{j=1}^{J} \frac{e^{(y_{ij}-\theta_i)^2/2\sigma_e^2}}{\sigma_e}$$

First, we set a_1 and a_2 to 0.001, b_1 and b_2 to 1000, μ_0 to 0, and σ_0^2 to 10^{10}, making the inverse gamma priors flat and uninformative. The results can be found in Table 3.8, Table 3.9 and Figure 3.5. We compare our estimates to the estimates from Gibbs samplers ran for 1.1 million iterations (which is a lot higher than the number of iterations in any of our algorithm runs in Table 3.8) with last 0.1 million iterations used as a sample. One small problem we found was that our model underestimated σ_θ^2 compared to the Gibbs sampler. This error is correctable if we apply tighter cutoffs for the $\hat{R}_c$ and $\hat{R}_{interval}$ of Gelman-Rubin diagnostics, to make our algorithm run longer. For consistency, we here let the cutoffs for the $\hat{R}_c$ and $\hat{R}_{interval}$ be the same as other examples. Other than that, our estimates and trace plots show that our model worked fine.

Table 3.8: Results of MCMC:VCM, flat inverse gamma priors. Results of 10 independent runs of full algorithm in Section 3.4.

	Estimates									
σ_θ^2	3687.3	4113.2	4039.6	3273.6	3907.1	3710.9	3291.5	3346.2	3714.8	3771.4
σ_e^2	2763.2	2832.4	2649.5	2729.7	2713.0	2773.8	2810.0	2779.7	2764.6	2760.1
μ	1527.9	1528.0	1529.7	1527.2	1529.8	1529.5	1527.0	1527.1	1527.3	1528.1
θ_1	1509.7	1509.1	1506.6	1508.9	1509.0	1509.0	1509.7	1510.2	1509.1	1509.8
θ_2	1527.4	1525.7	1528.4	1528.7	1527.0	1527.3	1529.8	1527.6	1528.3	1529.0
θ_3	1556.1	1556.4	1558.3	1556.9	1558.2	1557.2	1557.1	1556.5	1555.8	1557.1
θ_4	1504.2	1505.0	1504.0	1505.2	1503.9	1503.9	1503.5	1504.1	1503.6	1502.7
θ_5	1586.4	1584.9	1587.3	1585.8	1585.7	1587.6	1584.8	1585.6	1585.6	1587.7
θ_6	1480.6	1482.3	1479.3	1481.6	1480.1	1481.2	1481.6	1481.5	1482.8	1479.8
	Acceptance Rates									
	0.1714	0.1891	0.1557	0.2001	0.1822	0.2203	0.2037	0.2135	0.1728	0.1413
	Runtime (in seconds)									
	37.27	28.69	20.36	21.28	21.55	21.22	35.58	52.48	24.66	20.09

Table 3.9: Summary Statistics for the Estimates in Table 3.8: VCM, flat inverse gamma priors. Results from a Gibbs sampler run are also presented, for comparison.

	Mean of Estimates	SD of Estimates	Min Estimate	Max Estimate	Gibbs
σ_θ^2	3685.6	299.3	3273.6	4113.2	3891.8
σ_e^2	2757.6	51.2	2649.5	2832.4	2769.1
μ	1528.2	1.1	1527.0	1529.8	1527.4
θ_1	1509.1	1.0	1506.6	1510.2	1509.5
θ_2	1527.9	1.2	1525.7	1529.8	1527.9
θ_3	1557.0	0.8	1555.8	1558.3	1556.8
θ_4	1504.0	0.7	1502.7	1505.2	1503.8
θ_5	1586.1	1.1	1584.8	1587.7	1585.6
θ_6	1481.1	1.1	1479.3	1482.8	1481.2

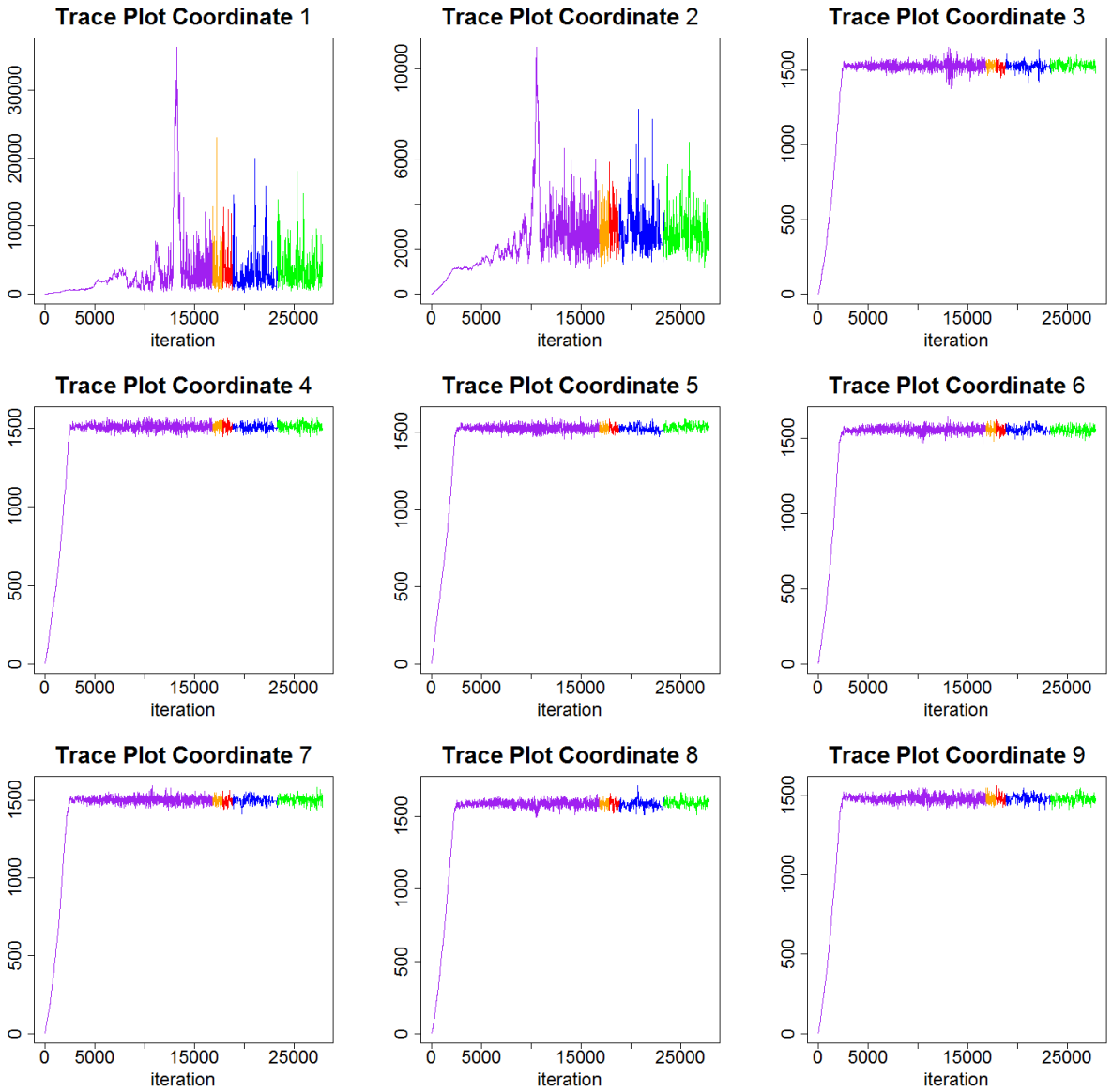


Figure 3.5: Trace Plots for Variance Components Model (with flat inverse gamma priors). Each coordinate corresponds to the parameters listed in Table 3.8 as ordered.

This time, we set a_1 and a_2 to 300, b_1 and b_2 to 100 while everything else remained same as before. This makes the inverse gamma priors really concentrated, and the results are shown in Table 3.10, Table 3.11 and Figure 3.6. As we can see from Figure 3.6, in this example, the transient phase was crucial to find the mode so in the 2^{nd} adaption phase we can avoid using 'bad' values to estimate the covariance of the target distribution.

Table 3.10: Results of MCMC:VCM, concentrated inverse gamma priors. Results of 10 independent runs of full algorithm in Section 3.4.

	Estimates									
σ_θ^2	3.5057	3.4959	3.4998	3.5326	3.5163	3.5169	3.5004	3.5144	3.5067	3.5008
σ_e^2	171.38	171.76	170.70	170.55	171.16	171.31	171.07	170.58	171.38	171.55
μ	1527.5	1527.3	1527.3	1527.7	1527.5	1527.9	1527.6	1527.4	1527.5	1527.7
θ_1	1525.3	1525.3	1525.3	1525.5	1525.4	1525.8	1525.4	1525.3	1525.3	1525.6
θ_2	1527.6	1527.5	1527.3	1527.6	1527.5	1527.8	1527.4	1527.3	1527.6	1527.7
θ_3	1531.0	1530.7	1530.7	1531.1	1531.0	1531.2	1530.8	1530.8	1531.0	1531.0
θ_4	1524.7	1524.8	1524.6	1524.8	1524.7	1525.1	1524.7	1524.4	1524.7	1524.9
θ_5	1534.2	1534.2	1534.0	1534.4	1534.3	1534.7	1534.1	1534.2	1534.3	1534.4
θ_6	1522.2	1522.1	1522.0	1522.1	1522.2	1522.7	1522.2	1521.7	1522.3	1522.3
	Acceptance Rates									
	0.2659	0.2631	0.2559	0.2520	0.2705	0.2858	0.2972	0.2635	0.2673	0.2816
	Runtime (in seconds)									
	18.33	17.28	18.57	18.20	24.06	23.33	20.54	16.83	21.69	14.16

Table 3.11: Summary Statistics for Estimates in Table 3.10: VCM, concentrated inverse gamma priors. Results from Gibbs sampler are also presented, for comparison.

	Mean of Estimates	SD of Estimates	Min Estimate	Max Estimate	Gibbs
σ_θ^2	3.5089	0.0111	3.4959	3.5326	3.5060
σ_e^2	171.14	0.42	170.55	171.76	171.08
μ	1527.5	0.2	1527.3	1527.9	1527.5
θ_1	1525.4	0.2	1525.3	1525.8	1525.4
θ_2	1527.5	0.2	1527.3	1527.8	1527.5
θ_3	1530.9	0.2	1530.7	1531.2	1530.8
θ_4	1524.7	0.2	1524.4	1525.1	1524.7
θ_5	1534.3	0.2	1534.0	1534.7	1534.2
θ_6	1522.2	0.2	1521.7	1522.7	1522.1

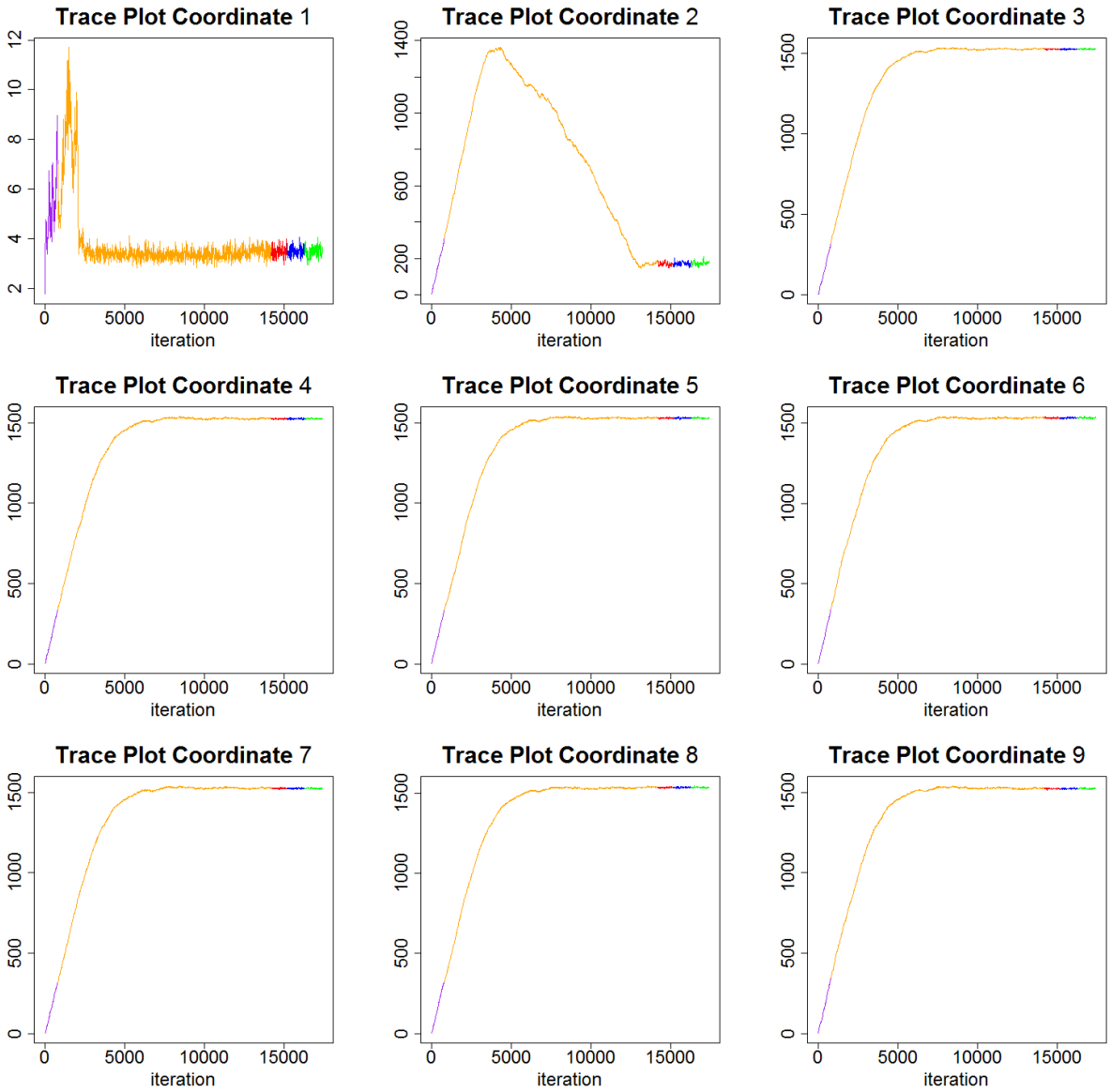


Figure 3.6: Trace Plots for Variance Components Model (with concentrated inverse gamma priors). Each coordinate corresponds to the parameter listed in Table 3.10 as ordered.

3.5.5 A Strongly Multimodal Example

The 'strongly multimodal' target density function we work with is

$$\frac{1}{3} * N(\mu_1, \Sigma\Sigma^t) + \frac{1}{3} * N(\mu_2, \Sigma\Sigma^t) + \frac{1}{3} * N(\mu_3, \Sigma\Sigma^t)$$

where each component of Σ is randomly drawn from $N(0, 1)$. The parameter values are

$$\begin{cases} \mu_1 &= (21.62166, -10.00424, 15.49878)^T \\ \mu_2 &= (9.671977, -28.515220, -12.744802)^T \\ \mu_3 &= (26.0518930, 0.2331812, -0.3433256)^T \end{cases}$$

$$\Sigma\Sigma^t = \begin{pmatrix} 1.2742983 & 0.1801673 & -1.353580 \\ 0.1801673 & 2.6300580 & 1.451527 \\ -1.3535803 & 1.4515267 & 4.861334 \end{pmatrix}.$$

We first ran 10 different chains with the starting values of the 1^{st} adaption phase chosen randomly from $U(-30, 30)$ for every coordinate of each chain. Since we want only one chain for one particular mode, once every chain ran for both 1^{st} adaption phase and transient phase, we reduced the number of chains down, based on the criteria described in Section 3.4.5, leaving us with 3 different chains. Then, we ran for the 2^{nd} adaption phases for all 3 chains, separately. Once this is done, we once again checked every chain had different modes with the values generated in the 2^{nd} adaption phase. Then we randomly chose 7 starting points as described in Section 3.4.5. With 10 different starting points in total (7 starting values randomly chosen and the last values from the 3 different chains), we created 10 replicative chains to run for the sampling phase. The plots for the sampling phase in Figure 3.7 are from one replicative chain which used the last values of the 2^{nd} adaption phase of the chain started with 'mode 1' as the starting point of the sampling phase. We did 10 different runs with the same starting values for the 1^{st} adaption phase, and the end

results we got from the runs are shown in Table 3.12, Table 3.13 and Figure 3.7.

Table 3.12: Results of MCMC: Multimodal Distribution. Results of 10 independent runs of full algorithm in Section 3.4.5.

	Estimates									
	18.764	18.170	20.413	18.965	18.853	19.746	19.690	18.687	19.241	20.200
μ	-13.440	-14.311	-9.723	-12.907	-13.093	-11.551	-11.769	-13.604	-12.467	-10.825
	0.9235	-0.1113	2.7895	0.3334	-0.0131	1.0996	1.8526	1.1745	0.9141	1.5964
	Acceptance Rates									
	0.3374	0.3489	0.3057	0.3449	0.3464	0.3370	0.3627	0.3472	0.3448	0.3471
	Runtime (in seconds)									
	72.04	71.40	70.26	72.93	70.31	74.74	71.33	72.50	72.20	71.29

Table 3.13: Summary Statistics for the Estimates in Table 3.12: Multimodal Distribution. μ is the true mean of target distribution.

	Mean of Estimates	SD of Estimates	Min Estimate	Max Estimate	μ
	19.273	0.719	18.170	20.413	19.115
	-12.369	1.401	-14.311	-9.723	-12.762
	1.0559	0.8827	-0.1113	2.7895	0.804

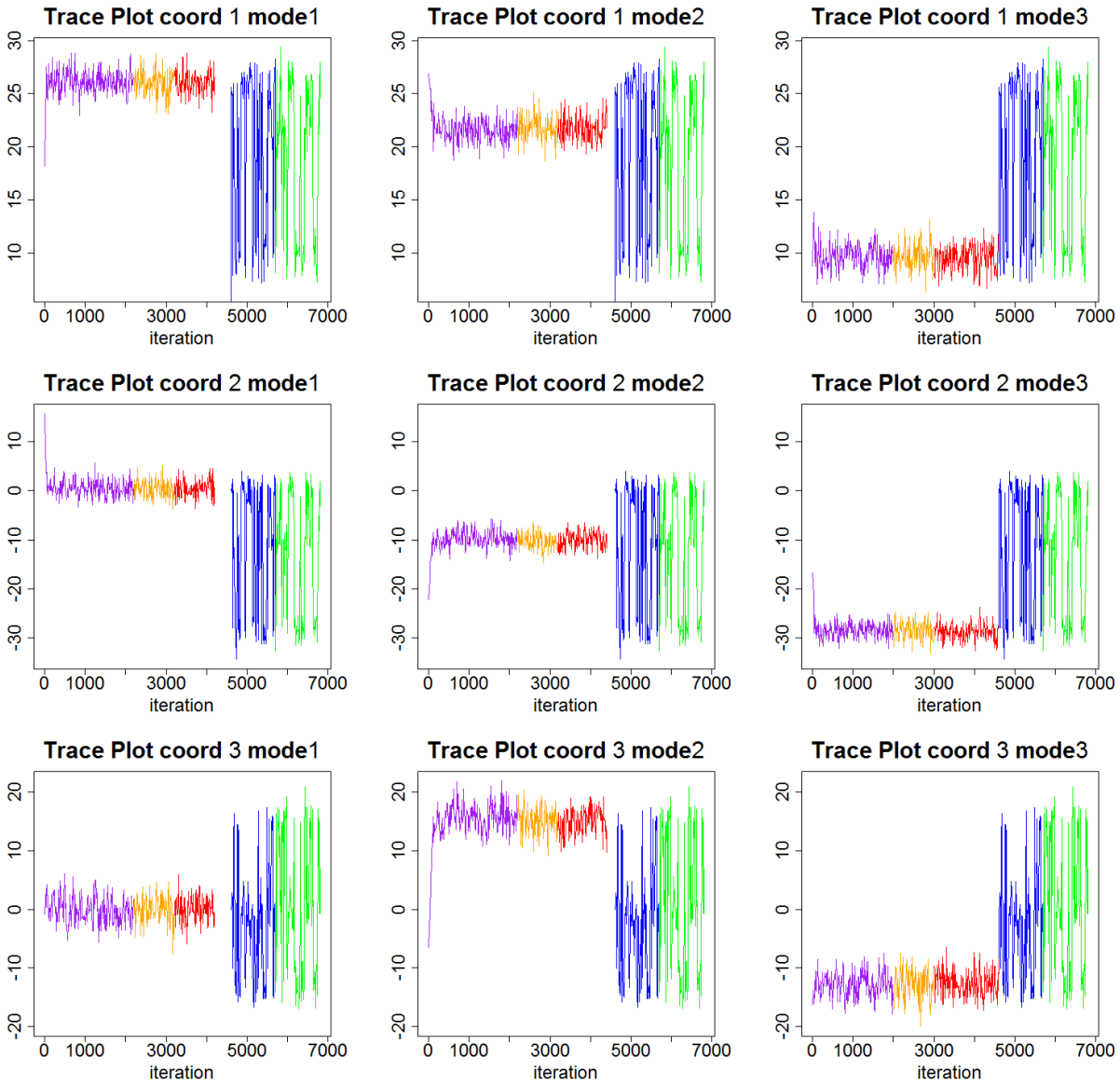


Figure 3.7: Trace Plots for Mixture of Three 3-Dimensional Multivariate Normals. Each row represents each coordinate and each column represents each chain trapped in different mode until the sampling phase.

As we see from Figure 3.7, the mixing of the Markov chain for the sampling phase looks to be good and does not just occur within one mode as in the previous phases.

3.6 Discussion

This chapter has presented a new algorithm, implemented in the R package 'atmcmc' (Yang 2014), to run adaptive MCMC for a finite amount of time, diagnose when the adaption is sufficient, and then run conventional MCMC with standard convergence diagnostics to get good target distribution convergence and estimates. The algorithm was illustrated on a number of examples, and found to perform quite well in each case.

We finish by discussing a number of related issues.

3.6.1 Acceptance Rate

Our algorithm makes heavy use of the fact (Roberts et al. 1997; Roberts and Rosenthal 2001) that for the symmetric random walk Metropolis algorithm on certain target densities, the optimal asymptotic acceptance rate is 0.234, which can be achieved if we use the proposal distribution $N(X_n, \Sigma_p)$ where $\Sigma_p = (2.38^2/d) * \Sigma$ and Σ is the target covariance matrix. Now, for certain truncated (discontinuous) target densities, the optimal acceptance rate is actually 0.1353 (Neal et al. 2012), and the true optimal value of the multiple to Σ is unknown. We used the multiple $2.38^2/d$ whether the target density is truncated or not. The acceptance rates we found in Table 3.1, Table 3.3 and Table 3.6 are slightly higher than the optimal acceptance rates, which is reasonable as our Markov chains have dimensions far from infinity. The acceptance rates for the Variance Components Model (VCM) look to be more puzzling, as they vary between 0.1353 and 0.234 in Table 3.8 but are close to 0.234 in Table 3.10. In VCM some coordinates are truncated and some are not, and there is little known about the optimal asymptotic acceptance rate in this case. Also, note that for the 'strongly multimodal' algorithm extension, the acceptance rates are a lot higher than 0.234 as seen in Table 3.12. One explanation for the high acceptance rate can be the direct jumps between the modes, but this only accounts for $0.05\,(= \alpha)$ at most.

3.6.2 Significance of Transient and 2^{nd} Adaption Phase

One question is whether our 2^{nd} adaption phase helps in terms of convergence speed, or whether our 1^{st} adaptive phase alone would be sufficient. To check this, we removed the 2^{nd} adaption phase from the full algorithm and ran each unimodal example three times, with starting values for Gelman-Rubin diagnostics picked based on the flat part of the transient phase. The runtimes we got for 3 different runs of the 9-dimensional multivariate normal example are 121.07, 79.89, and 76.13 seconds, respectively. These runtimes are all larger than any of runtimes we got for our full MCMC model in Table 3.1. Estimates from all three runs came out reasonable. For the logistic regression example, we got 8.78, 6.84 and 9.43 seconds for 3 runs; for pump failure data, we got 5330.87, 5007.72 and 1490.13 seconds; for VCM with flat inverse gamma priors, we got 72.65, 149.96 and 80.34 seconds, and for VCM with concentrated inverse gamma priors, we got 202.66, 89.36 and 153.84 seconds. Estimates from all these runs came out reasonably good except σ_θ^2 for VCM with flat inverse gamma priors was even more underestimated (about 2700 - 3200) than results in Table 3.9 from full algorithm. We conclude that runtimes were larger (most times by a lot) without a 2^{nd} adaption phase than with our full algorithm.

As another test, we tried removing both the transient phase and the 2^{nd} adaption phase, and again ran each unimodal example three times. (Here the starting values for the Gelman-Rubin diagnostic were chosen based on the values from the 1^{st} adaption phase, since we had neither a transient phase nor a 2^{nd} adaption phase; that is, we had to pick starting values for the sampling phase without having a rough idea on the range of target distribution.) For the 9-dimensional multivariate normal example, we got runtimes of 101.70, 80.22 and 69.03 seconds; for logistic regression, we got 8.47, 8.19 and 7.53 seconds; for pump failure data, we got 10756.04, 2593.72 and 2866.11 seconds; for VCM with flat inverse gamma priors, we got 6227.88, 744.50 and 72.83 seconds, and for VCM with concentrated inverse gamma priors,

we got 918.97, 2774.98 and 2551.30 seconds. The estimates again came out reasonable. But once again, all runtimes were larger than the corresponding ones for our full algorithm.

3.6.3 Comparison with a Full-dimensional Metropolis

We next consider how our algorithm fares against a full-dimensional Metropolis algorithm. Notice that we are not comparing our algorithm to the full-dimensional Metropolis with a really good proposal distribution. Finding a good proposal distribution for the full-dimensional Metropolis algorithm is the goal of the first three phases of our algorithm, and if we know a good proposal distribution from the start, there is no need to adapt the chain. However, in most cases of MCMC examples, we have little to no idea on the target distribution, and picking a good proposal distribution out of all possible choices is virtually impossible without adaption.

Since our algorithm runs replicative chains in the sampling phase to check for the convergence, it wouldn't be fair to compare our algorithm to one full-dimensional Metropolis chain in terms of runtime. Also, if we only run one full-dimensional Metropolis chain, we have to find a way to figure out when the full-dimensional Metropolis chain achieved the same level of convergence as the chain by our algorithm. Thus, we compare our algorithm with the full-dimensional Metropolis algorithm in terms of number of iterations they run for, and we run the same number of replicative chains for the full-dimensional Metropolis algorithm to check for the convergence through Gelman-Rubin diagnostics, using the same $\hat{R}_c$ and $\hat{R}_{interval}$ cutoffs with our algorithm.

We ran the full-dimensional Metropolis on the pump failure data from Section 3.5.3 and the VCM from Section 3.5.4. The proposal distributions for the Metropolis algorithm are $N(X_n, I)$ for the pump failure data and the VCM with flat inverse gamma priors and $N(X_n, 10I)$ for the VCM with concentrated inverse gamma priors, where X_n is the current

state value and I is the identity matrix. We took the initial value $0.1 * \mathbf{1}$ for the Metropolis algorithm, which is the initial value of the runs of our algorithm for all unimodal examples in Section 3.5. We also ran the same number of replicative chains, 10, as what we used in the runs of our algorithm in Section 3.5 for the Gelman-Rubin diagnostics, with the same $\hat{R}_c$ and $\hat{R}_{interval}$ cutoffs. With pump failure data, we ran the full-dimensional Metropolis for 2 million iterations, but the algorithm failed to achieve the convergence by the Gelman-Rubin diagnostics. Our algorithm was proved to be significantly more efficient in this case in terms of number of iterations since all of the runs in Section 3.5.3 converged in between 81200 and 126200 iterations. For VCM with flat inverse gamma priors, the Metropolis algorithm again didn't achieve the convergence for 2 million iterations by the Gelman-Rubin diagnostics. All the runs of our algorithm for the same example, shown in Section 3.5.4, achieved the convergence in between 156800 and 299600 iterations. For VCM with concentrated inverse gamma priors, the Metropolis algorithm achieved the convergence in 1775200 iterations, and the comparative runs in Section 3.5.4 converged somewhere in between 77200 and 210200 iterations. Thus, we conclude that our algorithm performed notably better in these particular cases.

Even if the proposal distribution is not as bad as what we just described, our algorithm still beat the full-dimensional Metropolis algorithm with a decent proposal distribution. When we removed both transient and 2^{nd} adaption phase from our algorithm in Section 3.6.2, we practically ran the full-dimensional Metropolis algorithm with a roughly-tuned diagonal proposal covariance matrix. So, we took the run with the fewest iterations for the 1^{st} adaption phase out of three runs from Section 3.6.2 to compare the efficiency of our algorithm to the full-dimensional Metropolis with a roughly tuned proposal distribution. This comparison is really not fair to our algorithm since we counted the number of iterations for the 1^{st} adaption phase against our algorithm but gave the tuned proposal distribution obtained from the 1^{st} adaption phase of our algorithm to the full-dimensional Metropolis

from the start. However, our algorithm still did better.

For pump failure data example, two chains with the fewest number of iterations for the 1^{st} adaption phase from Section 3.6.2 took 664200 and 333800 iterations, respectively, counting only for the sampling phase, or in other words the standard full-dimensional Metropolis phase. Thus, the full-dimensional Metropolis algorithm took 2.65 to 8.18 times more iterations to reach the same level of the convergence compared to the runs in Section 3.5.3. For VCM with flat inverse gamma priors, the number of iterations for the run with the fewest 1^{st} adaption period from Section 3.6.2 was 547400. Our algorithm ran 1.83 to 3.49 times faster in terms of number of iterations. For VCM with the concentrated inverse gamma priors, our algorithm was 1.58 to 4.57 times better than the full-dimensional Metropolis run from Section 3.6.2.

3.6.4 Initial Value for a Markov Chain

One important issue for improving Markov chain convergence speed is getting a good starting value. This problem is solved naturally for unimodal target distributions in our algorithm since we go through all the different phases, including the mode-finding transient phase, before starting the final sampling phase. The information we obtain in the first 3 phases are used to get a rough idea about the target distribution, and hence to pick good starting values for the sampling phase. The situation is a bit more problematic for a 'strongly multimodal' target distribution. There it is important to have well-dispersed starting points to find all the modes in the target distribution, right from the beginning. If we fail to find some modes, our resulting estimates will be compromised. Of course, similar concerns apply to virtually all MCMC algorithms and diagnostic approaches.

Implementing MCMC in practice does have some difficulties depending on the target dis-

tribution one is dealing with. We hope that the algorithm presented in this chapter, and implemented in the R package 'atmcmc' (Yang 2014), will provide a simple and efficient approach that can be applied to most target distributions which arise in practice.

Chapter 4

Ergodicity of Combocontinuous Adaptive MCMC Algorithms

4.1 Introduction

Markov Chain Monte Carlo (MCMC) algorithms are very widely used to analyze complex probability distributions (see e.g. Brooks et al. 2011). Adaptive MCMC algorithms adjust the Markov chain transition probabilities on the fly, in an attempt to improve efficiency, based on the past and/or current information from the chain. Adaptive MCMC algorithms can be quite effective in practice (see e.g. Haario et al. 2001, 2006; Roberts and Rosenthal 2009; Giordani and Kohn 2010; Vihola 2012; Turro et al. 2007), but the chain usually loses the Markovian property so that convergence to the target (stationary) distribution is no longer guaranteed (see e.g. Rosenthal 2004). Many papers present conditions which assure this convergence (e.g. Haario et al. 2001, 2006; Giordani and Kohn 2010; Vihola 2012; Atchadé and Rosenthal. 2005; Andrieu and Moulines 2006; Andrieu and Atchadé 2007; Roberts and Rosenthal 2007; Fort et al. 2011), but these conditions are usually difficult to verify in practice. By contrast, the results of Craiu et al. (2015) provide more easily checkable conditions

that guarantee convergence of adaptive MCMC algorithms, however they require continuity of all of the transition densities which makes their application somewhat limited in practice.

It was shown in Roberts and Rosenthal (2007) that the convergence of an adaptive MCMC algorithm is implied by the two conditions of Diminishing Adaption and Containment (explained herein). In practice, Diminishing Adaption is easily satisfied simply by constructing the adaptive mechanism appropriately. Unfortunately, the Containment condition is a lot harder to establish (see e.g. Bai et al. 2011b). Craiu et al. (2015) introduced several simple assumptions about upper and lower bounds on transition densities which, assuming continuity, guarantee the Containment condition and thus make ergodicity much easier to verify.

In this chapter, we relax one of the assumptions in Craiu et al. (2015) which is required to guarantee the Containment of an adaptive MCMC algorithm. The results of Craiu et al. (2015) require the transition kernel densities (or proposal kernel densities for the Metropolis-Hastings algorithm) of an adaptive MCMC algorithm to be continuous in x, y and γ. We here show that the joint continuity assumption on the kernel densities can be relaxed to a weaker assumption which we call "combocontinuity", for x and γ jointly, which includes the usual piecewise-continuity assumption as a special case, and which allows for e.g. truncated densities. We prove our result by generalising Dini's Theorem (about uniform convergence of compactly-supported functions) to the combocontinuous case, and then applying that theorem to the case of combocontinuous transition densities.

Below, Sections 4.2 and 4.3 present background about Adaptive MCMC and about combocontinuity. Section 4.4 presents a special case of our new algorithm, and Sections 4.5 and 4.6 present our general result. This result is proved in Section 4.9, together with Section 4.7 which generalises Dini's Theorem, and Section 4.8 which proves two lemmas about combocontinuity. Finally, Section 4.10 illustrates our results with two numerical examples of adaptive Metropolis-Hastings algorithms with combocontinuous proposal kernel densities.

4.2 Background about Adaptive MCMC

Consider a general state space $\mathcal{X}$ with a corresponding Borel σ-algebra $\mathcal{F}$, on which is defined a target probability distribution π. Suppose that for each γ in some index set $\mathcal{Y}$, P_γ is a valid MCMC algorithm, i.e. a time-homogeneous Markov chain kernel which leaves π stationary and is Harris ergodic so that $\lim_{n\to\infty} \|P_\gamma^n(x,\cdot) - \pi(\cdot)\| = 0$ for each fixed $x \in \mathcal{X}$. (Here $\|P_\gamma^n(x,\cdot) - \pi(\cdot)\| := \sup_{A\in\mathcal{F}} |P_\gamma^n(x,A) - \pi(A)|$ is the total variation distance to the target distribution π after n iterations of P_γ.)

An *adaptive* MCMC algorithm $\{X_n\}$ uses some specified rule to select an index value Γ_n at each iteration, based on current and/or past information from the chain and/or auxiliary randomness. It then updates X_n according to the Markov kernel P_{Γ_n}, so that for each $x \in \mathcal{X}$ and $A \in \mathcal{F}$,

$$\mathbf{P}[X_{n+1} \in A \mid X_n = x, \Gamma_n = \gamma, X_0, \ldots, X_{n-1}, \Gamma_0, \ldots, \Gamma_{n-1}] \;=\; P_\gamma(x, A)\,.$$

If the adaption rule is chosen wisely, to attempt to achieve some sort of optimality, then adaptive MCMC algorithms sometimes provide very dramatic speed-ups in efficiency and convergence to stationarity (e.g. Roberts and Rosenthal 2009). However, allowing Γ_n to depend on previous values of the $\{X_n\}$ can introduce biases so that the limiting distribution of X_n, if it exists at all, might be quite different than π (cf. Rosenthal 2004, and Example 4 of Roberts and Rosenthal 2007). This raises the question of what conditions assure convergence in distribution of $\{X_n\}$ to π, i.e. ensure that

$$\lim_{n\to\infty} \sup_{A\in\mathcal{F}} |\mathbf{P}(X_n \in A) - \pi(A)| = 0. \tag{4.1}$$

There have been many recent results about convergence of adaptive MCMC, as mentioned in the Introduction. Here we focus on the theorem of Roberts and Rosenthal (2007) which

states that the convergence of an adaptive MCMC algorithm is ensured by two conditions: Diminishing Adaption and Containment. Diminishing Adaption requires the algorithm to adapt less and less as the chain moves along, or more formally that

$$\lim_{n\to\infty} \sup_{x\in\mathcal{X}} \|P_{\Gamma_{n+1}}(x,\cdot) - P_{\Gamma_n}(x,\cdot)\| \;=\; 0 \text{ in probability}\,. \tag{4.2}$$

Containment requires the convergence times of the algorithm to remain bounded in probability, or more formally that for all $\epsilon > 0$,

$$\{M_\epsilon(X_n,\Gamma_n)\}_{n=1}^{\infty} \text{ is bounded in probability}\,, \tag{4.3}$$

where $M_\epsilon(x,\gamma) := \inf\{n \geq 1 : \|P_\gamma^n(x,\cdot) - \pi(\cdot)\| \leq \epsilon\}$ is the time required to get to within ϵ of the stationary distribution π when beginning at the state x and proceeding according to the fixed Markov chain kernal P_γ. Now, since the adaptive rule can be specified by the user, satisfying Diminishing Adaption is usually straightforward to ensure. On the other hand, the Containment condition is often difficult to verify in practice, requiring substantial specialised effort (e.g. Bai et al. 2011b).

4.3 Combocontinuous Functions

Craiu et al. (2015) introduced several simple assumptions about upper and lower bounds on transition densities which, assuming strong continuity conditions, guarantee the Containment condition and thus make ergodicity much easier to verify. However, the required continuity conditions are inconvenient. For one simple example, if using a truncated normal distribution (as is often used in this context), it is not permitted to use a "firm" truncation (Figure 4.1(a)), but rather it is necessary to linearly interpolate the truncation (Figure 4.1(b)), which is not difficult but which requires additional programming to implement. The present chapter

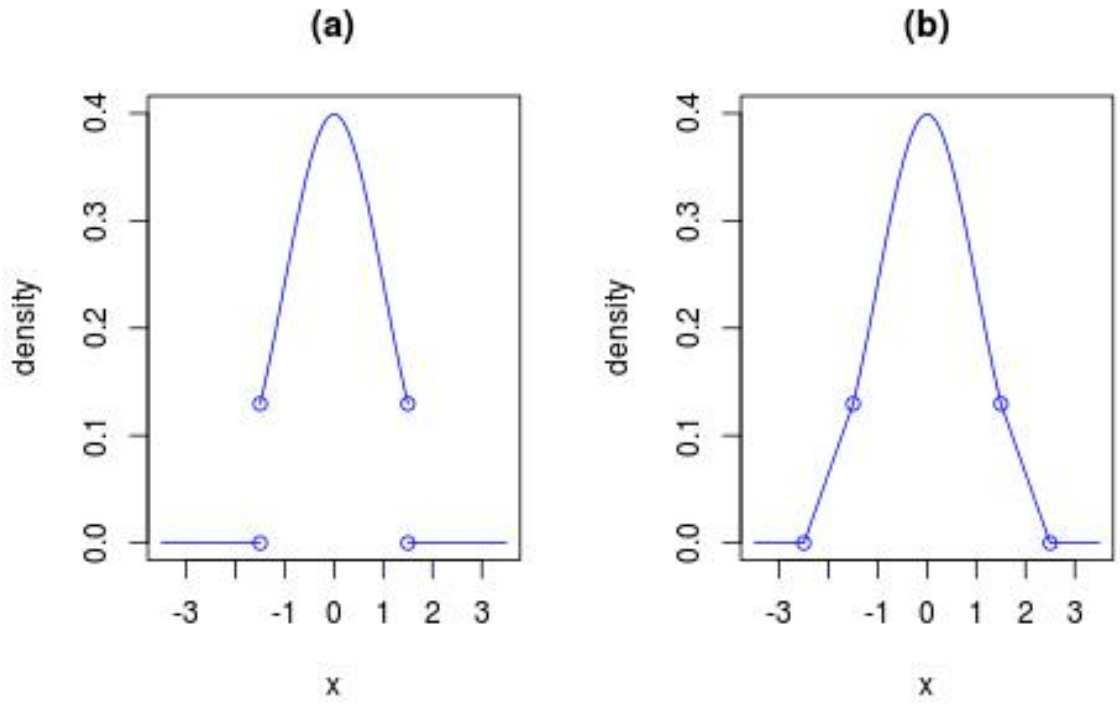


Figure 4.1: Two ways of truncating a normal density: with (a) a "firm" truncation (left), or (b) a "linear" truncation (right).

avoids this challenge by allowing for a more general notation of "combocontinuous" functions, a generalisation of piecewise-continuous functions.

We define a function f on a space S to be *combocontinuous* if it can be written as a finite combination of continuous functions, i.e. if $f(x) = g_{I(x)}(x)$ for some $m \in \mathbb{N}$, some (measurable) index function $I : S \to \{1, 2, \ldots, m\}$, and some finite collection $g_1, g_2, \ldots, g_m$ of continuous functions on S.

If I is constant on intervals, e.g. $I(x) = 1$ for $a \leq x \leq b$ and $I(x) = 2$ for $b < x \leq c$, then combocontinuity reduces to the usual notion of piecewise-continuity. In particular, firm truncations (as in Figure 4.1(a)) are always combocontinuous. On the other hand, if desired, a combocontinuous function could be much more complicated than a piecewise continuous function; for example, if $I(x) = 1$ for rational x, and $I(x) = 2$ for irrational x, then different proposals will be used from rational and from irrational states. In this chapter, we mostly focus on the case of truncated densities.

Note that combocontinuous functions share many properties of continuous functions. For example, if the space S is compact, then a combocontinuous function f must be bounded above and below (since each g_i is), and if each g_i is positive then also $\inf_{x \in S} f(x) > 0$.

4.4 The Bounded Adaption Metropolis (BAM) Algorithm

To illustrate our results in a simple but useful case, consider the following Bounded Adaption Metropolis (BAM) Algorithm, which is an easier-to-implement version of the similarly-named algorithm presented in Craiu et al. (2015).

Let $\mathcal{X} = \mathbb{R}^d$, let $K \subseteq \mathcal{X}$ be a (large) bounded region, let π be a continuous positive density on $\mathcal{X}$, and let $D > 0$ be a (large) constant. Let $\mathcal{Y}$ be any compact collection of

d-dimensional positive-definite matrices, and fix some specific $\Sigma_* \in \mathcal{Y}$. (One can set K and D really large, so in the practice, these bounds most likely won't interfere with the chain. However, here, we want the theoretical result which covers every situation, so we set the bounds in K and D.)

Define a process $\{X_n\}$ as follows: $X_0 = x_0$ for some fixed $x_0 \in K$. Then for $n = 0, 1, 2, \ldots$, given X_n, we generate a proposal Y_{n+1} by: (a) if $X_n \notin K$, then $Y_{n+1} \sim N(X_n,\ \Sigma_*)$; or (b) if $X_n \in K$, then $Y_{n+1} \sim N(X_n,\ \Sigma_{n+1})$, where the matrix $\Sigma_{n+1} \in \mathcal{Y}$ is selected in some fashion, perhaps depending on X_n and on the chain's entire history. Once Y_{n+1} is chosen, then if $|Y_{n+1} - X_n| > D$, the proposal is rejected so $X_{n+1} = X_n$. Otherwise, if $|Y_{n+1} - X_n| \leq D$, then with probability

$$a(X_n, Y_{n+1}) = \begin{cases} \min[1,\ \frac{\pi(Y_{n+1})}{\pi(X_n)}] & \text{if } X_n \in K \,\&\, Y_{n+1} \in K \text{ or } X_n \notin K \,\&\, Y_{n+1} \notin K \\ \min[1,\ \frac{\pi(Y_{n+1}) q_{\Sigma_*}(Y_{n+1}, X_n)}{\pi(X_n) q_{\Sigma_{n+1}}(X_n, Y_{n+1})}] & \text{if } X_n \in K \,\&\, Y_{n+1} \notin K \\ \min[1,\ \frac{\pi(Y_{n+1}) q_{\Sigma_{n+1}}(Y_{n+1}, X_n)}{\pi(X_n) q_{\Sigma_*}(X_n, Y_{n+1})}] & \text{if } X_n \notin K \,\&\, Y_{n+1} \in K \end{cases}$$

the proposal is accepted so $X_{n+1} = Y_{n+1}$, or with the remaining probability the proposal is rejected so $X_{n+1} = X_n$. (Note that $q_\Sigma(x, y)$ is the transition density from x to y with a covariance matrix Σ.)

Special case For example, Σ_{n+1} could be chosen to be $(2.38)^2 V_n / d$ where V_n is the empirical covariance matrix of $X_0, \ldots, X_n$ from the process's previous history (except restricted to some compact set $\mathcal{Y}$), since that choice approximates the optimal proposal covariance, cf. Haario et al. (2001); Roberts and Rosenthal (2009). One way to define V_n is let $V_n = \text{Cov}(\langle X_0 \rangle, \langle X_1 \rangle, \ldots, \langle X_n \rangle) + \epsilon I_d$ for some arbitrarily small constant $\epsilon > 0$. $\langle X_i \rangle$ is a shrunken version of X_i, i.e. $\langle X_i \rangle_j = \max(-L, \min(L, X_{i,j}))$ for some (large) constant $L > 0$, with j indexing for j^{th} coordinate. This idea of defining V_n is from Section 12.3 of Craiu et al. (2015).

Lemma 5. *If an $n \times n$ positive-definite real matrix A satisfies $c_1 I_n \leq A \leq c_2 I_n$ where $0 < c_1 < c_2$ are real numbers, then each entry of A is less than or equal to c_2.*

Proof. Since A is positive-definite, it is unitarily equivalent to a diagonal matrix D, and we still have $c_1 I_n \leq D \leq c_2 I_n$. It follows that the eigenvalues of D (and hence also of A) are all between c_1 and c_2, and also that the norm of D (and hence also A) is at most c_2. Now, if we let e_i be the unit vector in the i^{th} coordinate, then the i, j entry of A is equal to (Ae_j, e_i). But then

$$|(Ae_j, e_i)| \leq \|A\|\|e_j\|\|e_i\| = \|A\| \leq c_2$$

□

Proposition 6. *Consider the 'special case' described above. Let $\mathcal{Y} = \{\gamma \mid \gamma$ is a $d \times d$ positive definite matrix, $\epsilon\, I_d \leq \gamma \leq (8L^2 + \epsilon) d\, I_d\}$. Then $\mathcal{Y}$ is compact and every V_n is in $\mathcal{Y}$. Also, the 'special case' of BAM algorithm satisfies the Diminishing Adaption condition, i.e.* (4.2)

Proof. By Lemma 5, γ is bounded in the Euclidean norm since each entry of γ is bounded by $(8L^2 + \epsilon)d$. Thus, $\mathcal{Y}$ is bounded, and it is also closed by definition. $\mathcal{Y}$ is then compact.

A sample covariance matrix is positive semi-definite, and adding $\epsilon\, I$ to a positive semi-definite matrix makes the matrix positive definite. Thus, V_n is positive definite. In the 'special case' described above, individual covariance is bounded above by $8L^2$ and below by $-8L^2$, and individual variance is bounded above by $8L^2 + \epsilon$ and below by ϵ. By Lemma 5, if $V_n \leq \delta I_d$ for some $\delta < \epsilon$, then every entry of V_n has to be less than or equal to δ. But the diagonal elements of V_n is bounded below by ϵ. Thus, $\epsilon\, I_d \leq V_n$. Also, all the eigenvalues are positive for a positive definite matrix and the sum of the eigenvalues is equal to the trace of the matrix, so the largest eigenvalue is bounded above $(8L^2 + \epsilon)d$ for V_n. As V_n is positive

definite and unitarily equivalent to a diagonal matrix D, it is bounded above by $(8L^2+\epsilon)d\, I_d$. Hence, $\epsilon\, I_d \leq V_n \leq (8L^2+\epsilon)d\, I_d$. Therefore, every $V_n \in \mathcal{Y}$.

Denote consecutive transition kernels for the 'special case' of BAM algorithm as P_{V_n} and $P_{V_{n+1}}$ respectively. Recall the recursive formula of Σ_n on page 12. For the 'special case', the formula can be rewritten as

$$V_{n+1} = \frac{n-1}{n} V_n + \frac{1}{n}(n\langle \overline{X}_n\rangle\langle \overline{X}_n\rangle^T - (n+1)\langle \overline{X}_{n+1}\rangle\langle \overline{X}_{n+1}\rangle^T + \langle X_n\rangle\langle X_n\rangle^T + \epsilon I_d)\,,$$

where $\langle \overline{X}_n\rangle = \frac{1}{n}\sum_{i=0}^{n-1}\langle X_i\rangle$. Then $P_{V_{n+1}} \to P_{V_n}$ as $n \to \infty$ since $V_{n+1} \to V_n$ as $n \to \infty$. Thus, the 'special case' of BAM algorithm satisfies the Diminishing Adaption condition, i.e. (4.2). □

For the BAM algorithm, our results herein prove that the Containment condition holds, and hence convergence to stationarity also holds assuming Diminishing Adaption:

Theorem 7. *The above BAM algorithm satisfies Containment* (4.3)*. Hence, if the selection of the* Σ_n *satisfies Diminishing Adaption* (4.2)*, then convergence to stationarity* (4.1) *holds.*

This stands in contrast to other situations in which it is very difficult or impossible to establish convergence of adaptive MCMC algorithms. Theorem 7 is proved in Section 4.9 below, as a special case of our more general results.

4.5 More General Conditions

The above BAM algorithm already has good flexibility to design efficient adaptive MCMC algorithms. However, our results prove convergence in much greater generality, subject to certain conditions as we now discuss.

Let $\mathcal{X}$ be a general state space, equipped with a Borel σ-algebra $\mathcal{F}$, and a metric η. Assume there is some "origin" point $0 \in \mathcal{X}$. Let P be a fixed transition kernel for a

time-homogeneous Markov chain on $\mathcal{X}$, which is Harris ergodic to a stationary probability distribution π. Consider a stochastic process $\{X_n\}$ on $\mathcal{X}$ with the following properties:

(a) The process $\{X_n\}$ never moves more than some fixed finite distance $D > 0$ in any one step, i.e.

$$\mathbf{P}\Big(\eta(X_{n+1} - X_n) > D\Big) \;=\; 0\,,$$

so in particular the kernel P satisfies that

$$P\Big(x, \{y \in \mathcal{X} : \eta(x,y) \le D\}\Big) = 1, \quad x \in \mathcal{X}.$$

(b) The process $\{X_n\}$ moves by the fixed transition kernel P whenever the current state $X_n = x$ is outside of a fixed compact subset $K \subset \mathcal{X}$, i.e. $\mathbf{P}[X_{n+1} \in A \,|\, X_n = x, X_{n-1}, \ldots, X_0] = P(x, A)$ for $x \notin K$. Inside of K, the process can move arbitrarily in a non-anticipating way, subject only to measurability, to anywhere within K_D, where K_r is defined to be the set of $\forall x \in \mathcal{X}$ within a distance $r > 0$ of K.

(c) The fixed kernel P is bounded above by $P(x, dy) \le M\mu_*(dy)$ for some finite constant $M > 0$, for all $x \in K_D \backslash K$ and all $y \in K_{2D} \backslash K_D$, where μ_* is any probability measure concentrated on $K_{2D} \backslash K_D$.

(d) The fixed kernel P is bounded below by $P^{n_0}(x, A) \ge \epsilon\nu_*(A)$ (P^{n_0} is a n_0-step transition probability.) for some probability measure ν_* on $\mathcal{X}$, some $n_0 \in \mathbb{N}$, and some constant $\epsilon > 0$, for all $x \in K_{2D} \backslash K_D$ and all $A \in \mathcal{F}$. ν_* must be either (1) $\nu_* = \mu_*$ or (2) ν_* can be any probability measure on $\mathcal{X}$ if P is reversible with respect to π and $\mu_* = \pi|_{K_{2D} \backslash K_D}$. ($\mu_*$ here is the μ_* in (c) above.)

(Note that in the case of a Metropolis-Hastings algorithm, if the corresponding proposal

kernels Q_γ and Q satisfy the assumptions (a), (b) and (c), then P_γ and P automatically satisfy them too.)

By Theorem 5 of Craiu et al. (2015), if a stochastic process $\{X_n\}$ satisfies the above conditions, then it is bounded in probability, i.e.

$$\lim_{L\to\infty} \sup_{n\in\mathbb{N}} \mathbf{P}(\eta(X_n, 0) > L \,|\, X_0 = x_0) \;=\; 0.$$

Furthermore, by Proposition 6 of Craiu et al. (2015), if $\mathcal{X}$ is an open subset of $\mathbb{R}^d$, then condition (d) above, with $\nu_* = \text{Uniform}(K_{2D}\backslash K_D)$, is implied by the following condition (d'):

(d') The fixed kernel P is bounded below by $P(x, dy) \geq \epsilon Leb(dy)$ (Leb is the Lebesgue measure) whenever $x, y \in J$ with $|y - x| < \delta$ for some $\epsilon > 0$ and $\delta > 0$, where J is any bounded rectangle with $J \supset K_{2D}\backslash K_D$.

We will give a special attention to a Metropolis-Hastings algorithm (Metropolis et al. 1953; Hastings 1970). The algorithm works as follows. At each iteration n, conditioned on the current state X_n, the Markov chain proposes Y_{n+1} from some proposal kernel $Q_\gamma(X_n, \cdot)$. The new proposal Y_{n+1} is accepted with probability

$$a(X_n, Y_{n+1}) = \min\big[1, \frac{\pi(Y_{n+1})q_\gamma(Y_{n+1}, X_n)}{\pi(X_n)q_\gamma(X_n, Y_{n+1})}\big],$$

or Y_{n+1} is rejected with probability $1 - a(X_n, Y_{n+1})$. If Y_{n+1} is accepted, $X_{n+1} = Y_{n+1}$, if not, $X_{n+1} = X_n$.

We introduce a sub-Metropolis-Hastings algorithm (see Roberts and Rosenthal 2008) to describe the BAM algorithm presented in Section 4.4. The sub-Metropolis-Hastings algo-

rithm has a transition kernel

$$P_\gamma(x, dy) = a(x, y)Q'_\gamma(x, dy) + r(x)\delta_x(dy),$$

where $r(x) = 1 - \int_{\mathcal{X}} a(x, y)Q'_\gamma(x, dy)$ and $\delta_x(\cdot)$ is a point-mass at x. With probability $\int_{\mathcal{X}} Q'_\gamma(x, dy) \leq 1$, it follows the usual Metropolis-Hastings algorithm with a proposal kernel $Q_\gamma(x, y) = Q'_\gamma(x, y)/\int_{\mathcal{X}} Q'_\gamma(x, dy)$. With probability $1 - \int_{\mathcal{X}} Q'_\gamma(x, dy)$, it stays at the current state, i.e.$X_{n+1} = X_n$. If a proposal kernel of a Metropolis-Hastings algorithm is truncated such as the BAM algorithm (e.g. Y_{n+1} rejected if $\eta(Y_{n+1}, X_n) > D$ for some constant $D > 0$), it is a sub-Metropolis-Hastings algorithm.

A ϕ-irreducible, full-dimensional Metropolis-Hastings algorithm is known to be Harris recurrent (see e.g. Tierney 1994; Roberts and Rosenthal 2006). A Markov chain on $(\mathcal{X}, \mathcal{F})$ is ϕ-irreducible if $\mathbf{P}(\tau_A < \infty | X_0 = x) > 0$ for all $x \in \mathcal{X}$ and all $A \in \mathcal{F}$ with $\psi(A) > 0$, for some non-zero, σ-finite measure ψ, where $\tau_A = \inf\{n \geq 1 : X_n \in A\}$. A Metropolis-Hastings algorithm with a centered normal proposal kernel (e.g. $N(X_n, \Sigma)$) is ϕ-irreducible since the chain can reach anywhere in the state space $\mathcal{X}$ no matter where the Markov chain starts. It is still ϕ-irreducible even though Y_{n+1} rejected if $\eta(Y_{n+1}, X_n) > D$ if the target density is everywhere positive. Thus, a full-dimensional Metropolis-Hastings algorithm with a centered truncated normal proposal kernel is Harris recurrent if the target density is everywhere positive. Indeed, for any reversible Markov chain with stationary distribution π (everywhere positive), if the transitions are truncated at some fixed distance, then π remains stationary, conditioned on that the truncated chain is still ϕ-irreducible.

To prove Containment for an adaptive MCMC algorithm, an additional condition is needed besides $\{X_n\}$ being bounded in probability. One way to proceed is in terms of density functions. We shall assume that with respect to some reference measure $\lambda(\cdot)$ on $\mathcal{X}$, π has a density g so that $\pi(dy) = g(y)\lambda(dy)$, and furthermore *either* each kernel P_γ has

a density p_γ with respect to $\lambda(\cdot)$ so $P_\gamma(x, dy) = p_\gamma(x, y)\lambda(dy)$, or each P_γ is a Metropolis-Hastings algorithm whose proposal kernel Q_γ has a density q_γ with respect to $\lambda(\cdot)$ so that $Q_\gamma(x, dy) = q_\gamma(x, y)\lambda(dy)$. We also assume the reference measure $\lambda(\cdot)$ gives finite measure to every bounded set and for any $x \in \mathcal{X}$ and any $D > 0$, $\lambda(\{y \in \mathcal{X} | \eta(x, y) = D\}) = 0$.

In terms of these assumed densities, we introduce an additional assumption (e) as follows.

(e) In terms of the above assumed densities and reference measure, g is a continuous positive density function for π. Furthermore, either:

(e1) The mapping $(x, \gamma) \mapsto p_\gamma(x, y)$ is continuous in γ and jointly combocontinuous in (x, γ) in the sense that: $p_\gamma(x, y) = \alpha_{\gamma, I(x)}(x, y)\mathbf{1}_{\eta(x,y) \le D}$ for some index function $I : \mathcal{X} \to \{1, 2, \ldots, m\}$ for some $m \in \mathbb{N}$. Here, $\alpha_{\gamma,i}(x, y)$ is jointly continuous in x and γ for each fixed y, and $\int_{\mathcal{X}} \alpha_{\gamma,i}(x, y)\, \lambda(dy) = 1$ for $i = 1, \ldots, m$. Assume $\alpha_{\gamma,i}(x, y)$ are uniformly bounded, thus $p_\gamma(x, y)$ is uniformly bounded.

(e2) Or, in the case of a Metropolis-Hastings algorithm: The proposal densities $(x, \gamma) \mapsto q_\gamma(x, y)$ is continuous in γ and jointly combocontinuous in (x, γ) and uniformly bounded in the same way as above (i.e., upon substituting $q_\gamma(x, y)$ for $p_\gamma(x, y)$ in the above conditions).

Remark. *If $\alpha_{\gamma,i}(x, y)$ are jointly combocontinuous in (x, y, γ), then $\alpha_{\gamma,i}(x, y)$ are uniformly bounded in a compact space and so is $p_\gamma(x, y)$ (or $q_\gamma(x, y)$).*

In Craiu et al. (2015), a version of assumption (e) was used in which λ was assumed to be Lebesgue measure, and full continuity was assumed in place of combocontinuity, thus leading to inconvenient application as discussed in Section 4.3 above. Out of two assumptions in (e), (e2) probably carries most of practical relevance. However, for the generality of our result, we also prove the case of (e1).

4.6 Main Result

In terms of the above conditions, we have the following theorem which guarantees Containment, and hence also convergence provided Diminishing Adaption is satisfied.

Theorem 8. *Consider an adaptive MCMC algorithm as above. If the algorithm satisfies the assumptions (a), (b), (c), (d), and (e), and if the space $\mathcal{Y}$ of Markov kernel indices is compact, then the algorithm satisfies the Containment condition* (4.3)*. Hence, if it also satisfies the Diminishing Adaption condition* (4.2)*, then it converges to stationarity as in* (4.1)*.*

Proof. First, by Theorem 5 of Craiu et al. (2015), we know that the process $\{X_n\}$ is bounded in probability since it satisfies conditions (a), (b), (c), and (d).

Next, it follows from Lemmas 10 and 11 in Section 4.8 below that for each $n \in \mathbb{N}$, the mapping

$$(x, \gamma) \mapsto \Delta(x, \gamma, n) := \|P_\gamma^n(x, \cdot) - \pi(\cdot)\| \tag{4.4}$$

is a jointly combocontinuous mapping in (x, γ), and each 'piece' is a non-increasing function of n which converges to 0. Then, by applying Theorem 9 (Generalised Dini's Theorem) in Section 4.7 below to the function $f_n(x, \gamma) = \Delta(x, \gamma, n + 1)$, we obtain that

$$\lim_{n\to\infty} \sup_{x\in\mathcal{C}} \sup_{\gamma\in\mathcal{Y}} \Delta(x, \gamma, n) = 0$$

for any compact set $\mathcal{C} \subset \mathcal{X}$.

The rest of the proof to show the Containment condition holds is the same as the last part of the proof of Proposition 23 in Craiu et al. (2015). To repeat here, if $\{X_n\}$ is bounded in probability, then for any $\delta > 0$, there is a compact subset $\mathcal{C}$ such that $P(X_n \notin$

$\mathcal{C}) \leq \delta$ for all n. With any $\epsilon > 0$, we can find n such that $\sup_{x\in\mathcal{C}} \sup_{\gamma\in\mathcal{Y}} \Delta(x, \gamma, n) < \epsilon$. Thus, $\sup_{x\in\mathcal{C}} \sup_{\gamma\in\mathcal{Y}} M_\epsilon(x, \gamma) < \infty$ for any $\epsilon > 0$.Let $L := \sup_{x\in\mathcal{C}} \sup_{\gamma\in\mathcal{Y}} M_\epsilon(x, \gamma)$. Then $P(M_\epsilon(X_n, \Gamma_n) > L) \leq \delta$ for all n. Therefore, the Containment condition holds, i.e. (4.3) is satisfied.

Finally, if the adaptive MCMC algorithm also satisfies the Diminishing Adaption condition, then by Roberts and Rosenthal (2007), the algorithm converges to π, (although we don't know how fast it converges), in total variation distance, i.e. (4.1) holds. □

It remains to prove the Generalised Dini's Theorem and the technical lemmas used in the proof of Theorem 8 above, which we do next.

4.7 Generalisation of Dini's Theorem

Dini's Theorem may be stated as follows (see e.g. Theorem 7.13 in Rudin 1976). Let $\{f_n\}$ be a sequence of continuous real-valued functions defined on a compact set C, which is non-decreasing (i.e. $f_n(x) \leq f_{n+1}(x)$ for each fixed n and $x \in C$), and which converges pointwise to a continuous function f (i.e. $\lim_{n\to\infty} f_n(x) = f(x)$ for each fixed $x \in \mathcal{X}$). Then the convergence is uniform, i.e. $\lim_{n\to\infty} \sup_{x\in C} |f_n(x) - f(x)| = 0$.

In this section, we generalise Dini's Theorem to the combocontinuous case, so that the theorem can be applied to prove Theorem 8.

Theorem 9. (Generalised Dini's Theorem)

Suppose a set C is compact, and $\{f_n\}$ is a sequence of real-valued functions on C, and f is a continuous real-valued function on C, and:

1. *For each $n \in \mathbb{N}$, f_n can be expressed as $f_n(z) = f_{n,I(z)}(z)$ for some index function $I(z) \in \mathcal{J} = \{1, 2, \ldots, m\}$, and some $m \in \mathbb{N}$, and some collection $f_{n,i}$ of functions.*

2. *Each of these $f_{n,i}$ is a continuous real-valued function on $\overline{C}_i$, the closure of the subset $C_i = \{z \in C \mid I(z) = i\}$.*

3. *For each $i \in \mathcal{J}$, $\{f_{n,i}\}$ converges pointwise to f on $\overline{C}_i$.*

4. *For each $i \in \mathcal{J}$, $f_{n,i}(z) \geq f_{n+1,i}(z)$ for all $z \in \overline{C}_i$, $n = 1, 2, 3, ...$*

Then $f_n \to f$ uniformly on C, i.e. $\lim_{n\to\infty} \sup_{x\in C} |f_n(x) - f(x)| = 0$.

Proof. This follows by applying the original Dini's Theorem separately on each subset $\overline{C}_i$. For a complete proof, let $g_{n,i} = f_{n,i} - f$ for each $i \in \mathcal{J}$. Let, for $z \in C$, $g_n(z) = g_{n,I(z)}(z) = f_{n,I(z)}(z) - f(z)$ with $I(x) \in \mathcal{J}$. Since $\{f_{n,i}\}$, $i \in \mathcal{J}$, converges pointwise to a continuous function f on $\overline{C}_i$, $\{g_{n,i}\}$, $i \in \mathcal{J}$ converges pointwise to 0 on $\overline{C}_i$. Also, for each $i \in \mathcal{J}$, since $f_{n,i}(z) \geq f_{n+1,i}(z)$ for all $z \in \overline{C}_i$, $g_{n,i} \geq g_{n+1,i}$ for all $z \in \overline{C}_i$.

Let $\epsilon > 0$ and $C_{n,i} = \{z \in \overline{C}_i | g_{n,i}(z) \geq \epsilon\}$, $i \in \mathcal{J}$. Then $C_{n,i}$ is closed, since $g_{n,i} = f_{n,i} - f$ is continuous on $\overline{C}_i$, and the continuous inverse image of any closed set is closed (e.g. Rudin, 1976, Theorem 4.8 Corollary). Hence, $C_{n,i}$ is compact, since closed subsets of compact sets are compact (e.g. Rudin, 1976, Theorem 2.35).

Next, note that $C_{n,i} \supset C_{n+1,i}$, since $g_{n,i} \geq g_{n+1,i}$ on $\overline{C}_i$. Pick $z \in \overline{C}_i$. Since $g_{n,i} \to 0$ on $\overline{C}_i$, $z \notin C_{n,i}$ if n is sufficiently large. Thus, for every $z \in \overline{C}_i$, we have that $z \notin \cap_{n=1}^{\infty} C_{n,i}$. It follows that $\cap_{n=1}^{\infty} C_{n,i}$ is the empty set. Hence, by the finite intersection property (e.g. Rudin, 1976, Theorem 2.36, Corollary), there must be some $N_i \in \mathbb{N}$ such that $C_{N_i,i}$ is empty.

Therefore, $0 \leq g_{n,i}(z) < \epsilon$ for all $z \in C_i$ and for all $n \geq N_i$. Hence, $0 \leq g_n(z) < \epsilon$ for all $z \in C$ and for all $n \geq \max(N_1, \ldots, N_m)$. Since ϵ is arbitrary, $f_n \to f$ uniformly on C. □

Remark. *In Theorem 9, it does* not *suffice to assume only that $f_{n,i}$ converges pointwise to f on C_i, i.e. the closure $\overline{C_i}$ really is required. For example, let $C = [0, 2]$, and $m = 2$, with $I(x) = 1$ for $x \in [0, 1)$ and $I(x) = 2$ for $x \in [1, 2]$. Then let $f_{n,1}(x) = x^n$, and*

$f_{n,2}(x) = f(x) = 0$. *Then* $f_{n,1} \to 0$ *pointwise on* $C_1 := [0, 1)$, *but* $\sup_{C_1} f_{n,1} = 1$ *for each* n, *so the convergence of* f_n *to* f *is not uniform.*

4.8 Lemmas About Combocontinuity

We here show that, under the assumptions of Theorem 8, the total variation distance mapping (4.4) is combocontinuous, and each 'piece' converges to 0. Then we can apply Theorem 9 (Generalisation of Dini's Theorem) to the mapping (4.4).

Lemma 10. *Consider an adaptive MCMC algorithm as in Section 4.2, with assumed densities as in Section 4.5. Assume conditions (a)and (e1). Then, for each* $n \geq 2$, *the function* $f_{n,\gamma}(x) := \|P_\gamma^n(x, \cdot) - \pi(\cdot)\|$ *is*

1. *jointly combocontinuous in* (x, γ) *in the sense that:* $f_{n,\gamma}(x) = f_{n,\gamma,I(x)}(x)$ *for some index function* $I : \mathcal{X} \to \{1, 2, \ldots, m\}$ *for some* $m \in \mathbb{N}$ *where* $f_{n,\gamma,i}(x)$ *is jointly continuous in* x *and* γ *for all* (x, γ), *and*

2. *for each fixed* (x, γ, i), $f_{n,\gamma,i}(x)$ *converges pointwise to 0 on* $\mathcal{X}$ *as* $n \to \infty$ *and is a non-increasing function in* n.

Proof. For ease of notation, we assume in condition (e1) that $m = 2$; the extension to larger m is then straightforward. By (e1), the mapping $(x, \gamma) \mapsto p_\gamma(x, y)$ is jointly combocontinuous in x and γ in the sense that:

$$p_\gamma(x, y) = \begin{cases} \alpha_{\gamma,1}(x, y), & \text{if } x \in S_1 \ \& \ \eta(x, y) \leq D \\ \alpha_{\gamma,2}(x, y), & \text{if } x \in S_2 \ \& \ \eta(x, y) \leq D \end{cases}$$

where $D > 0$ is some (large) constant; $\{S_1, S_2\}$ is a partition of $\mathcal{X}$; $\alpha_{\gamma,i}(x, y)$ is jointly continuous in x and γ; and $\int_{\mathcal{X}} \alpha_{\gamma,1}(x, y)\, \lambda(dy) = \int_{\mathcal{X}} \alpha_{\gamma,2}(x, y)\, \lambda(dy) = 1$. Recall that, by (a),

$p_\gamma(x,y) = 0$ when $\eta(x,y) > D$.

First, we prove a n-step transition probability density $p^n_\gamma(x,y)$ is continuous in γ and uniformly bounded. Define a set $A^n_{(x,y,D)}$ as

$$A^n_{(x,y,D)} = \{(z_1,\ldots,z_n) \in \mathcal{X}^n | \eta(x,z_1) \le D,\ \eta(z_{j-1},z_j) \le D,$$
$$\eta(z_j,z_{j+1}) \le D,\ \eta(z_n,y) \le D,\ j = 2,\ldots,n-1\}.$$

$A^n_{(x,y,D)}$ is the set of sequences of length n of states in $\mathcal{X}$ which is placed in between x and y and never moves more than a distance D. With $A^n_{(x,y,D)}$ defined above, for $n \ge 2$, a n-step transition probability density $p^n_\gamma(x,y)$ of a Markov chain can be written as

$$p^n_\gamma(x,y) = \int_{A^{n-1}_{(x,y,D)}} \cdots \int p_\gamma(x,y_1)p_\gamma(y_1,y_2)\ldots p_\gamma(y_{n-1},y)\,\lambda(dy_1)\ldots\lambda(dy_{n-1}) \tag{4.5}$$

By (e1), $p_\gamma(x,y)$ is continuous in γ and uniformly bounded. Also, $\lambda(A^{n-1}_{(x,y,D)})$ is finite. Then, $p^n_\gamma(x,y)$ is continuous in γ by the Bounded Convergence Theorem, and $p^n_\gamma(x,y)$ is uniformly bounded.

Second, we prove $p^n_\gamma(x,y)$ is jointly combocontinuous in x and γ and uniformly bounded. We write $p^n_\gamma(x,y)$ as

$$p^n_\gamma(x,y) = \begin{cases} h_{n,\gamma,1}(x,y), & \text{if } x \in S_1 \\ h_{n,\gamma,2}(x,y), & \text{if } x \in S_2. \end{cases}$$

We rearrange (4.5) as

$$p^n_\gamma(x,y) = \int_{\{y_1 \in \mathcal{X} | \eta(x,y_1) \le D\}} p_\gamma(x,y_1)p^{n-1}_\gamma(y_1,y)\,\lambda(dy_1)$$

Then, for $n \geq 2$,

$$h_{n,\gamma,1}(x,y) = \int_{\{y_1 \in \mathcal{X} | \eta(x,y_1) \leq D\}} \alpha_{\gamma,1}(x,y_1) p_\gamma^{n-1}(y_1,y) \lambda(dy_1)$$

Fix (x_0, γ_0). Let a sequence $\{(x_k, \gamma_k)\} \to (x_0, \gamma_0)$, $k = 1, \ldots$. Notice that $h_{n,\gamma,1}(x,y) = 0$ if $\eta(x,y) > nD$. Hence, we only prove the continuity and the uniform-boundness of $h_{n,\gamma,1}(x,y)$ when restricted to the subset $\{(x,y) : \eta(x,y) \leq nD\}$.

Suppose a Markov chain moves from the current state (either x_k or x_0) to the next state, $y_1 \in \mathcal{X}$. Group possible y_1's into

$$\begin{aligned}
U_1 &= \{y_1 \in \mathcal{X} | \eta(x_k,y_1) \leq D \ \& \ \eta(x_0,y_1) \leq D\} \\
U_2 &= \{y_1 \in \mathcal{X} | \eta(x_k,y_1) > D \ \& \ \eta(x_0,y_1) > D\} \\
U_3 &= \{y_1 \in \mathcal{X} | \eta(x_k,y_1) \leq D \ \& \ \eta(x_0,y_1) > D\} \\
U_4 &= \{y_1 \in \mathcal{X} | \eta(x_k,y_1) > D \ \& \ \eta(x_0,y_1) \leq D\}
\end{aligned}$$

Note that $\alpha_{\gamma_k,1}(x_k,y_1) = \alpha_{\gamma_0,1}(x_0,y_1) = 0$ for $y_1 \in U_2$; $\alpha_{\gamma_0,1}(x_0,y_1) = 0$ for $y_1 \in U_3$; and $\alpha_{\gamma_k,1}(x_k,y_1) = 0$ for $y_1 \in U_4$. Also, $\lambda(U_3) \to 0$ and $\lambda(U_4) \to 0$ as $k \to \infty$.

Then, for $n \geq 2$, as $k \to \infty$, by the Bounded Convergence Theorem,

$$\begin{aligned}
&|h_{n,\gamma_k,1}(x_k,y) - h_{n,\gamma_0,1}(x_0,y)| \\
&\leq \Big| \int_{y_1 \in U_1} \alpha_{\gamma_k,1}(x_k,y_1) p_{\gamma_k}^{n-1}(y_1,y) \, \lambda(dy_1) - \int_{y_1 \in U_1} \alpha_{\gamma_0,1}(x_0,y_1) p_{\gamma_0}^{n-1}(y_1,y) \, \lambda(dy_1) \Big| \\
&\qquad + \Big| \int_{y_1 \in U_3} \alpha_{\gamma_k,1}(x_k,y_1) p_{\gamma_k}^{n-1}(y_1,y) \, \lambda(dy_1) \Big| + \Big| \int_{y_1 \in U_4} \alpha_{\gamma_0,1}(x_0,y_1) p_{\gamma_0}^{n-1}(y_1,y) \, \lambda(dy_1) \Big| \\
&\to 0
\end{aligned}$$

since $\alpha_{\gamma,1}(x,y_1)$ is jointly continuous in x and γ and $p_\gamma^{n-1}(y_1,y)$ is continuous in γ and doesn't

depend on x; $\alpha_{\gamma,1}(x,y)$ and $p_\gamma^{n-1}(y_1,y)$ are uniformly bounded; and $\lambda(U_1)$ is finite. Thus, $h_{n,\gamma,1}(x,y)$ is jointly continuous in x and γ and uniformly bounded. Same proof applies for $h_{n,\gamma,2}(x,y)$. Therefore, $p_\gamma^n(x,y)$ is jointly combocontinuous in x and γ and uniformly bounded.

Third, we prove $\|P_\gamma^n(x,\cdot)-\pi(\cdot)\|$ is jointly combocontinuous in x and γ. By definition,

$$\|P_\gamma^n(x,\cdot)-\pi(\cdot)\| = 0.5 * \int_{\mathcal{X}} |p_\gamma^n(x,y)-g(y)|\lambda(dy) \tag{4.6}$$

We can also write (4.6) as

$$\begin{aligned}\|P_\gamma^n(x,\cdot)-\pi(\cdot)\| &= \begin{cases} 0.5 * \int_{\mathcal{X}} |h_{n,\gamma,1}(x,y)-g(y)|\lambda(dy), & \text{if } x \in S_1 \\ 0.5 * \int_{\mathcal{X}} |h_{n,\gamma,2}(x,y)-g(y)|\lambda(dy), & \text{if } x \in S_2 \end{cases} \\ &= \begin{cases} f_{n,\gamma,1}(x), & \text{if } x \in S_1 \\ f_{n,\gamma,2}(x), & \text{if } x \in S_2. \end{cases}\end{aligned} \tag{4.7}$$

Again, fix (x_0,γ_0). Let a sequence $\{(x_k,\gamma_k)\} \to (x_0,\gamma_0)$, $k=1,\ldots$. We mentioned above $h_{n,\gamma,1}(x,y)=0$ if $\eta(x,y)>nD$. Similarly as before, we group y's into

$$\begin{aligned} U_1^n &= \{y \in \mathcal{X} | \eta(x_k,y) \le nD \ \& \ \eta(x_0,y) \le nD\} \\ U_2^n &= \{y \in \mathcal{X} | \eta(x_k,y) > nD \ \& \ \eta(x_0,y) > nD\} \\ U_3^n &= \{y \in \mathcal{X} | \eta(x_k,y) \le nD \ \& \ \eta(x_0,y) > nD\} \\ U_4^n &= \{y \in \mathcal{X} | \eta(x_k,y) > nD \ \& \ \eta(x_0,y) \le nD\} \end{aligned}$$

Then, for $n \geq 2$, as $k \to \infty$, by the Bounded Convergence Theorem,

$$\begin{aligned}
&\Big| \int_{\mathcal{X}} |h_{n,\gamma_k,1}(x_k, y) - g(y)|\lambda(dy) - \int_{\mathcal{X}} |h_{n,\gamma_0,1}(x_0, y) - g(y)|\lambda(dy) \Big| \\
&\leq \Big| \int_{U_1^n} |h_{n,\gamma_k,1}(x_k, y) - g(y)|\lambda(dy) - \int_{U_1^n} |h_{n,\gamma_0,1}(x_0, y) - g(y)|\lambda(dy) \Big| \\
&\quad + \Big| \int_{U_3^n} |h_{n,\gamma_k,1}(x_k, y) - g(y)|\lambda(dy) \Big| + \Big| \int_{U_4^n} |h_{n,\gamma_0,1}(x_0, y) - g(y)|\lambda(dy) \Big| \\
&\quad + \Big| \int_{U_2^n} |g(y)|\lambda(dy) - \int_{U_2^n} |g(y)|\lambda(dy) \Big| \\
&\to 0
\end{aligned}$$

since $h_{n,\gamma,1}(x, y)$ is jointly continuous in x and γ on the subset $\{(x, y) : \eta(x, y) \leq nD\}$; $h_{n,\gamma,1}(x, y)$ is uniformly bounded; and $\lambda(U_1^n)$ is finite. Thus, $f_{n,\gamma,1}(x)$ from (4.7) is jointly continuous in x and γ. Same proof applies for $f_{n,\gamma,2}(x)$. Therefore, $\|P_\gamma^n(x, \cdot) - \pi(\cdot)\|$ is jointly combocontinuous in x and γ.

Lastly, we want to prove $\{f_{n,\gamma,1}\}$ and $\{f_{n,\gamma,2}\}$ from (4.7) converge to 0 on $\mathcal{X}$ and are non-increasing functions in n. Note that we need $\{f_{n,\gamma,1}\}$ and $\{f_{n,\gamma,2}\}$ from (4.7) to converge to 0 on $\mathcal{X}$ not just on S_1 and S_2, respectively. (This is because when we apply Theorem 9 (Generalisation of Dini's Theorem), we need the convergence on $\overline{S}_1$ and $\overline{S}_2$, respectively, not just on S_1 and S_2.) Hence, we need more than just stating P_γ is Harris ergodic to π. With P_γ, only $\{x \in S_1\}$ ($\{x \in S_2\}$) feeds into the function $f_{n,\gamma,1}(x)$ ($f_{n,\gamma,2}(x)$).

Notice that, for each fixed γ, $h_{n,\gamma,i}(x, y)$ is a n-step transition probability density of a Markov chain. This chain moves by a fixed transition kernel P_γ after 1^{st} iteration. No matter which state on $\mathcal{X}$ the chain jumps to by the 1^{st} iteration, which should be within the distance D from the initial state, from that point it converges to its stationary distribution π since P_γ is Harris ergodic to π. Hence, $\{f_{n,\gamma,1}\}$ and $\{f_{n,\gamma,2}\}$ from (4.7) with each fixed (x, γ) converge to 0 on $\mathcal{X}$ and are non-increasing functions in n for $n \geq 2$ (e.g. Proposition 3(c) of Roberts

and Rosenthal 2004). The result follows. □

Next, we consider a Metropolis-Hastings algorithm with combocontinuous proposal kernel densities as defined in Section 4.5.

Lemma 11. *Consider an adaptive Metropolis-Hastings algorithm algorithm as in Section 4.2, with assumed densities as in Section 4.5. Assume conditions (a)and (e2). Then, for each* $n \geq 2$*, the function* $f_{n,\gamma}(x) := \|P_\gamma^n(x,\cdot) - \pi(\cdot)\|$ *is*

1. *jointly combocontinuous in* (x,γ) *in the sense that:* $f_{n,\gamma}(x) = f_{n,\gamma,I(x)}(x)$ *for some index function* $I : \mathcal{X} \to \{1,2,\ldots,m\}$ *for some* $m \in \mathbb{N}$ *where* $f_{n,\gamma,i}(x)$ *is jointly continuous in* x *and* γ *for all* (x,γ)*, and*

2. *for each fixed* (x,γ,i)*,* $f_{n,\gamma,i}(x)$ *converges pointwise to 0 on* $\mathcal{X}$ *as* $n \to \infty$ *and is a non-increasing function in* n*.*

Proof. For ease of notation, we assume in condition (e2) that $m = 2$; the extension to larger m is then straightforward. By (e2), the mapping $(x,\gamma) \mapsto q_\gamma(x,y)$ is jointly combocontinuous in x and γ in the sense that:

$$q_\gamma(x,y) = \begin{cases} \beta_{\gamma,1}(x,y), & \text{if } x \in T_1 \;\&\; \eta(x,y) \leq D \\ \beta_{\gamma,2}(x,y), & \text{if } x \in T_2 \;\&\; \eta(x,y) \leq D \end{cases} \tag{4.8}$$

where $D > 0$ is some (large) constant; $\{T_1, T_2\}$ is a partition of $\mathcal{X}$; $\beta_{\gamma,i}(x,y)$ is jointly continuous in x and γ; and $\int_{\mathcal{X}} \beta_{\gamma,1}(x,y)\,\lambda(dy) = \int_{\mathcal{X}} \beta_{\gamma,2}(x,y)\,\lambda(dy) = 1$. Recall that, by (a), $q_\gamma(x,y) = 0$ when $\eta(x,y) > D$.

Let $a_\gamma(x)$ be the acceptance probability of a proposal from $x \in \mathcal{X}$ in this Metropolis-

Hastings algorithm. We write $a_\gamma(x)$ as

$$a_\gamma(x) = \int_{\{y \in \mathcal{X} | \eta(x,y) \leq D\}} \min\left[1, \frac{g(y)q_\gamma(y,x)}{g(x)q_\gamma(x,y)}\right] q_\gamma(x,y)\lambda(dy)$$

By (e2), $q_\gamma(x,y)$ is continuous in γ and uniformly bounded. Thus, by the Bounded Convergence Theorem, $a_\gamma(x)$ is continuous in γ. The transition kernel $P_\gamma(x, dy)$ of this algorithm can be written as

$$P_\gamma(x, dy) = [1 - a_\gamma(x)]\delta_x(dy) + w_\gamma(x,y)\lambda(dy).$$

where $\delta_x(\cdot)$ is a point-mass at x and $w_\gamma(x,y) = q_\gamma(x,y)\min\left[1, \frac{g(y)q_\gamma(y,x)}{g(x)q_\gamma(x,y)}\right]$. Notice that $w_\gamma(x,y)$ is continuous in γ since $q_\gamma(x,y)$ is continuous in γ. The n-step transition kernel $P^n_\gamma(x, dy)$ is then

$$P^n_\gamma(x, dy) = [1 - a_\gamma(x)]^n\delta_x(dy) + w^n_\gamma(x,y)\lambda(dy)$$

where we define

$$w^n_\gamma(x,y) = \sum_{S \neq \emptyset} w^{n,S}_\gamma(x,y).$$

Here the sum is over all non-empty subsets $S \subseteq \{1, 2, ..., n\}$, and $w^{n,S}_\gamma(x,y)$ is the sub-density corresponding to getting from x to y in n steps while accepting moves only at the times in S (while rejecting moves at all times not in S). For example, if $n = 5$ and $S = \{2, 4, 5\}$, then $w^{5,\{2,4,5\}}_\gamma(x,y)$ corresponds to transitioning from x to y in 5 steps, while the first and third proposals are rejected and the others are accepted. The transition density $w^{5,\{2,4,5\}}_\gamma(x,y)$ can

be thus written as

$$w_{\gamma}^{5,\{2,4,5\}}(x,y) = \iint\limits_{A^2_{(x,y,D)}} [1-a_{\gamma}(x)]w_{\gamma}(x,y_1)[1-a_{\gamma}(y_1)]w_{\gamma}(y_1,y_2)w_{\gamma}(y_2,y)\,\lambda(dy_1)\lambda(dy_2)$$

First, we prove $[1-a_{\gamma}(x)]^n$ is jointly combocontinuous in x and γ . We write

$$a_{\gamma}(x) = \begin{cases} a_{\gamma,1}(x), & \text{if } x \in T_1 \\ a_{\gamma,2}(x), & \text{if } x \in T_2. \end{cases}$$

Then, $a_{\gamma,i}(x)$ can be written as

$$a_{\gamma,i}(x) = \int_{\{y\in\mathcal{X}|\eta(x,y)\le D\}} \min\Big[1, \frac{g(y)q_{\gamma}(y,x)}{g(x)\beta_{\gamma,i}(x,y)}\Big]\beta_{\gamma,i}(x,y)\lambda(dy) \tag{4.9}$$

for $i = 1, 2$. Fix (x_0, γ_0). Let a sequence $\{(x_k, \gamma_k)\} \to (x_0, \gamma_0)$, $k = 1, \ldots$. Group possible y's into

$$U_1 = \{y \in \mathcal{X} | \eta(x_k, y) \le D \;\&\; \eta(x_0, y) \le D\}$$
$$U_2 = \{y \in \mathcal{X} | \eta(x_k, y) > D \;\&\; \eta(x_0, y) > D\}$$
$$U_3 = \{y \in \mathcal{X} | \eta(x_k, y) \le D \;\&\; \eta(x_0, y) > D\}$$
$$U_4 = \{y \in \mathcal{X} | \eta(x_k, y) > D \;\&\; \eta(x_0, y) \le D\}$$

Similarly as in the proof of Lemma 10, when $k \to \infty$, $\lambda(U_3), \lambda(U_4) \to 0$; $a_{\gamma,1}(x) = 0$ on U_2; $\min\Big[1, \frac{g(y)q_{\gamma}(y,x)}{g(x)\beta_{\gamma,1}(x,y)}\Big]\beta_{\gamma,1}(x,y)$ is jointly continuous in x and γ for each fixed y and uniformly bounded when restricted to the subset $\{(x,y) : \eta(x,y) \le D\}$; $\lambda(U_1)$ is finite. Thus, $a_{\gamma_k,1}(x_k) \to a_{\gamma_0,1}(x_0)$ as $k \to \infty$ by the Bounded Convergence Theorem; i.e. $a_{\gamma,1}(x)$ is jointly continuous in x and γ and so is $[1-a_{\gamma,1}(x)]$ and $[1-a_{\gamma,1}(x)]^n$. Same proof applies

for $[1 - a_{\gamma,2}(x)]^n$. We conclude that $[1 - a_\gamma(x)]^n$ is jointly combocontinuous in x and γ.

Second, we prove $w_\gamma^{n,S}(x, y)$ is jointly combocontinuous in x and γ and uniformly bounded when restricted to the subset $\{(x, y) : \eta(x, y) \leq lD\}$ where $|S| = l$. We write $w_\gamma^{n,S}(x, y)$ as

$$w_\gamma^{n,S}(x,y) = \begin{cases} w_{\gamma,1}^{n,S}(x,y), & \text{if } x \in T_1 \\ w_{\gamma,2}^{n,S}(x,y), & \text{if } x \in T_2\,. \end{cases} \tag{4.10}$$

The proof is similar to that of Lemma 10, but this time y_1 is the first proposal accepted after the current state x.

Fix (x_0, γ_0). Let a sequence $\{(x_k, \gamma_k)\} \to (x_0, \gamma_0)$, $k = 1, \ldots$. We group possible $y_1's$ into

$$\begin{aligned} U_1 &= \{y_1 \in \mathcal{X} | \eta(x_k, y_1) \leq D \;\&\; \eta(x_0, y_1) \leq D\} \\ U_2 &= \{y_1 \in \mathcal{X} | \eta(x_k, y_1) > D \;\&\; \eta(x_0, y_1) > D\} \\ U_3 &= \{y_1 \in \mathcal{X} | \eta(x_k, y_1) \leq D \;\&\; \eta(x_0, y_1) > D\} \\ U_4 &= \{y_1 \in \mathcal{X} | \eta(x_k, y_1) > D \;\&\; \eta(x_0, y_1) \leq D\}. \end{aligned}$$

Similarly as before, $|w_{\gamma_k,i}^{n,S}(x_k, y) - w_{\gamma_0,i}^{n,S}(x_0, y)| \to 0$ as $k \to \infty$ since: $\lambda(U_3), \lambda(U_4) \to 0$ as $k \to \infty$; $\alpha_{\gamma_k,i}(x_k, y) = \alpha_{\gamma_0,i}(x_0, y)$ for $y_1 \in U_2$; $[1 - a_\gamma(x)]$ and $w_\gamma(x, y)$ are continuous in γ and jointly continuous in x and γ; $\lambda(U_1)$ is finite so that we can apply the Bounded Convergence Theorem for the set U_1. Thus, $w_{\gamma,i}^{n,S}(x, y)$ is jointly continuous in x and γ and uniformly bounded. Therefore, $w_\gamma^{n,S}(x, y)$ is jointly combocontinuous in x and γ and uniformly bounded when restricted to the subset $\{(x, y) : \eta(x, y) \leq lD\}$ for every $l \leq n$.

Third, we prove $\|P_\gamma^n(x, \cdot) - \pi(\cdot)\|$ is jointly combocontinuous in x and γ. We write $\|P_\gamma^n(x, \cdot) - \pi(\cdot)\|$ as

$$\|P_\gamma^n(x,\cdot) - \pi(\cdot)\| = [1 - a_\gamma(x)]^n + 0.5 * \int_{\mathcal{X}} |w_\gamma^n(x, y) - g(y)| \lambda(dy). \tag{4.11}$$

We rewrite (4.11) as

$$
\begin{aligned}
&\|P_\gamma^n(x,\cdot)-\pi(\cdot)\| \\
&= \begin{cases} [1-a_{\gamma,1}(x)]^n + 0.5 * \int_{\mathcal{X}} |w_{\gamma,1}^n(x,y)-g(y)|\lambda(dy), & \text{if } x \in T_1 \\ [1-a_{\gamma,2}(x)]^n + 0.5 * \int_{\mathcal{X}} |w_{\gamma,2}^n(x,y)-g(y)|\lambda(dy), & \text{if } x \in T_2 \end{cases} \\
&= \begin{cases} f_{n,\gamma,1}(x), & \text{if } x \in T_1 \\ f_{n,\gamma,2}(x), & \text{if } x \in T_2. \end{cases}
\end{aligned} \tag{4.12}
$$

Since $w_{\gamma,i}^{n,S}(x,y)$ is jointly continuous in x and γ and uniformly bounded when restricted to the subset $\{(x,y) : \eta(x,y) \leq lD\}$ for every $l \leq n$, $w_{\gamma,i}^n(x,y)$ is jointly combocontinuous in x and γ. Now we prove the joint combocontinuity for the latter part of (4.11). The proof is essentially the same as the proof of Lemma 10 except this time y is grouped based on the distance lD not nD for every $l \leq n$, creating $4n$ groups of y. Thus, similarly as we proved in Lemma 10, by Bounded Convergence Theorem, the latter part of (4.11) is combocontinuous in x and γ.

Lastly, we want to prove $\{f_{n,\gamma,1}\}$ and $\{f_{n,\gamma,2}\}$ from (4.12) converge to 0 on $\mathcal{X}$ and are non-increasing functions in n. Same with the last part of the proof of Lemma 10, we want to prove $\{f_{n,\gamma,1}\}$ and $\{f_{n,\gamma,2}\}$ from (4.12) converge to 0 on $\mathcal{X}$, not just on T_1 and T_2, respectively. To prove this, we apply the same logic as in the proof of Lemma 10. $\{f_{n,\gamma,1}\}$ and $\{f_{n,\gamma,2}\}$ are the total variation distance of a Metropolis-Hastings algorithm, which moves by a fixed transition kernel $Q_\gamma(x,\cdot)$ after 1^{st} iteration. No matter which state the chain is at after 1^{st} iteration (which is within the distance of D from the initial state), from that point it converges to its stationary distribution π. Thus, both $\{f_{n,\gamma,1}\}$ and $\{f_{n,\gamma,2}\}$ with each fixed (x,γ) converge to 0 on $\mathcal{X}$ and are non-increasing functions in n for $n \geq 2$. The result follows. □

4.9 Proof of Theorem 7

Denote the density of π as g with respect to Lebesgue measure. Denote the proposal kernels for $x \in K$ as $\{Q^*_\gamma(x,\cdot)\}_{\gamma\in\mathcal{Y}}$ and the proposal kernel for $x \notin K$ as $Q(x,\cdot)$. We then define Q_γ as

$$Q_\gamma(x,\cdot) = \begin{cases} Q^*_\gamma(x,\cdot) & x \in K \\ Q(x,\cdot) & x \notin K \end{cases}$$

Since we reject a proposal y from x if $|x-y| > D$, $Q_\gamma(x,dy) = 0$ if $|x-y| > D$. Let $P_\gamma(x,\cdot)$ be a corresponding transition kernel for $Q_\gamma(x,\cdot)$.

The BAM algorithm with a fixed proposal kernel Q_γ is reversible with respect to π. Thus, π is a stationary distribution for the algorithm. As noted in Section 4.5, a full-dimensional Metropolis-Hastings algorithm with a centered truncated normal proposal kernel is Harris recurrent. Thus, the BAM algorithm with a fixed proposal kernel Q_γ is Harris recurrent. Since it is also aperiodic, it follows that the BAM algorithm with each fixed Q_γ is Harris ergodic to π.

As constructed, the BAM algorithm determines γ (or Σ_{n+1}) of $\{Q_\gamma(x,\cdot)\}_{\gamma\in\mathcal{Y}}$ at each iteration n based on the past and present states from the Markov chain. And $\mathcal{Y}$ is compact. It follows that the BAM algorithm follows the set-up of Section 4.2.

We also need to check if the algorithm satisfies the assumptions (a), (b), (c), (d) and (e) from Section 4.5. As noted in Section 4.5, we only need the corresponding Q_γ and Q to satisfy the assumptions (a), (b) and (c), to ensure that P_γ and P also satisfy them. The proposal kernels of the above BAM algorithm have bounded jumps since $Q_\gamma(x,dy) = 0$ when $|x-y| > D$. Thus, it satisfies the assumption (a). The chain moves by a fixed transition kernel, Q, outside of a compact subset K, satisfying the assumption (b). The fixed proposal kernel Q is bounded above as described in (c) since it is a normal distribution.

The state space $\mathcal{X}$ of the algorithm is an open subset of $\mathbb{R}^d$, so we can use the assumption (d') to imply the assumption (d). Since Q is a normal distribution and π is continuous on $\mathcal{X}$, the assumption (d') is satisfied. We can easily see that the proposal kernel densities $q_\gamma(x, y)$ of $Q_\gamma(x, dy)$ is continuous in γ and jointly combocontinuous in x and γ as in (e2).

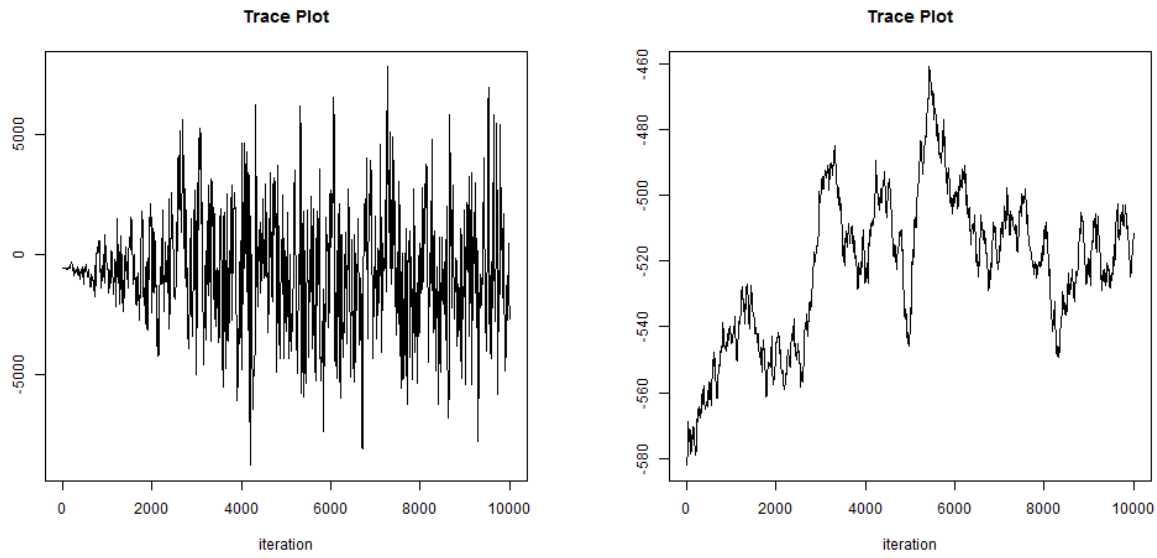
Therefore, by Theorem 8, the BAM algorithm satisfies Containment (4.3), thus proving Theorem 7. □

4.10 Numerical examples

In this section, we run the 'special case' of BAM algorithm ($\epsilon = 0.001$) described in Section 4.4 above, on two specific statistical examples, and compare its performances with those of non-adaptive Metropolis algorithms. The first example is in dimension $d = 9$, and the second one is in dimension $d = 12$. In both cases, our compact set K is defined as the d-dimensional cube $[-100000, 100000]^d$, and our step size bound is $D = 100000$. The fixed proposal kernel for the BAM algorithm when $X_n \notin K$ is $Q \sim N(X_n, I_d)$.

4.10.1 Application: 9-dimensional Multivariate Normal Distribution

We first consider the target distribution is $N(\mu, \Sigma)$, where $\mu \in \mathbb{R}^d$ with $d = 9$, and $\Sigma \in \mathbb{R}^{d\times d}$ are fixed and arbitrary (subject to Σ being positive-definite; in fact we set $\Sigma = M\,M^t$ where the matrix M was generated with iid normal entries). The starting value for the MCMC algorithm is μ itself. The trace plots (of coordinate 7) for a BAM algorithm and a standard Metropolis algorithm (with proposal kernal $N(X_n, I_d)$) are shown in Figure 4.2.



Bounded Adaption Metropolis

Standard Metropolis, $N(X_n, I_d)$

Figure 4.2: Trace plots (of coordinate 7) for a Bounded Adaption Metropolis (left) versus a Standard Metropolis algorithm with proposal kernel $N(X_n, I_d)$ (right), on a 9-dimensional multivariate normal target distribution, showing the superiority of the BAM algorithm.

We can see the mixing of the BAM algorithm is a lot better than the standard Metropolis. In the trace plots of the BAM algorithm, we see an increase in the average jumping distance from one state to the next state in the first a few thousands iterations, which implies the adaption was indeed effective.

4.10.2 Application: Pump Failure Model

We next consider a BAM algorithm for a true Bayesian statistical model, applied to the number of failures of pumps at a nuclear power plant. This model was first introduced by Gaver and O'Muircheartaigh (1987); we use the slightly different set-up from George et al. (1993). The resulting posterior density is

$$f(\lambda_1, ..., \lambda_n, \alpha, \beta | y_1, ..., y_n) \propto e^{-\alpha} \beta^{0.1-1} e^{-\beta} \prod_{i=1}^{n} \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} (\lambda_i t_i)^{y_i} e^{-\lambda_i t_i}.$$

Table 4.1: Pump Failure Data

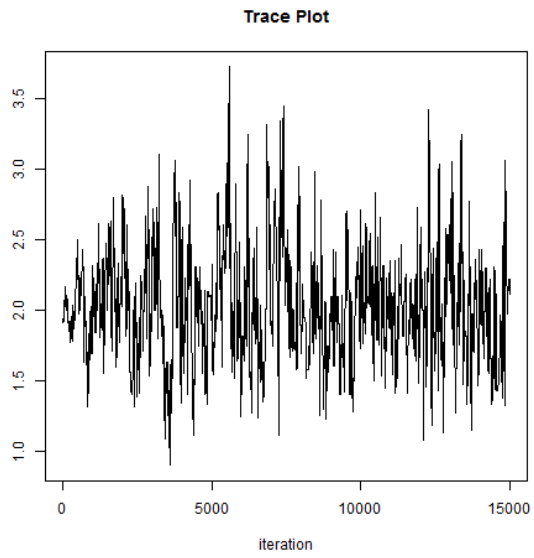
Obs. no.	1	2	3	4	5	6	7	8	9	10
y_i	5	1	5	14	3	19	1	1	4	22
t_i	94.320	15.720	62.880	125.760	5.240	31.440	1.048	1.048	2.096	10.480

This follows from the assumption that the number of failures follow the distribution

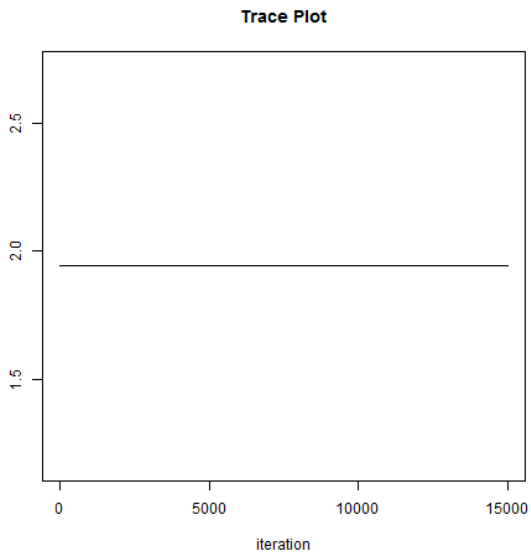
$$f(y_1, ..., y_n|\lambda_1, ..., \lambda_n) = \prod_{i=1}^{n} \text{Poisson}(\lambda_i t_i),$$

where λ_i is the failure rate of pump i, t_i is the length of operation time (in thousands of hours), and n is the number of pumps, and furthermore $\lambda_i \sim \text{Gamma}(\alpha, \beta)$, $\alpha \sim \text{Exp}(1)$, and $\beta \sim \text{Gamma}(0.1, 1)$. Here $n = 10$, so since $\lambda_1, ..., \lambda_n, \alpha, \beta > 0$, the state space is $(0, \infty)^d$ with dimension $d = n + 2 = 12$. The data for the model are the values of the y_i and t_i, which are reproduced in Table 4.1 herein.

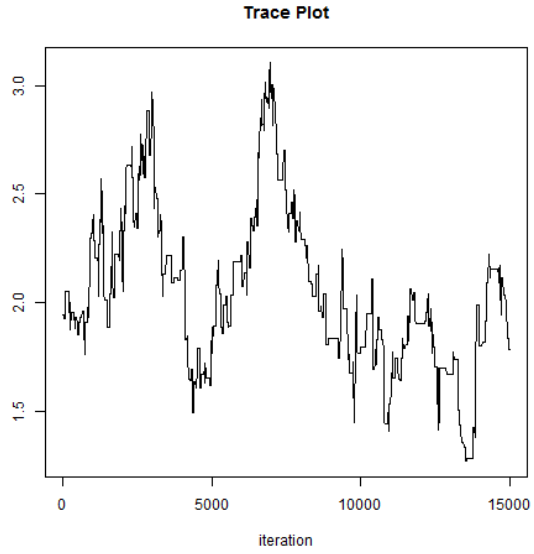
We run both Bounded Adaption Metropolis and standard Metropolis algorithm to compare them. For ease of comparison, each run uses initial values given by the estimates of each parameter obtained from a previous standard MCMC run.



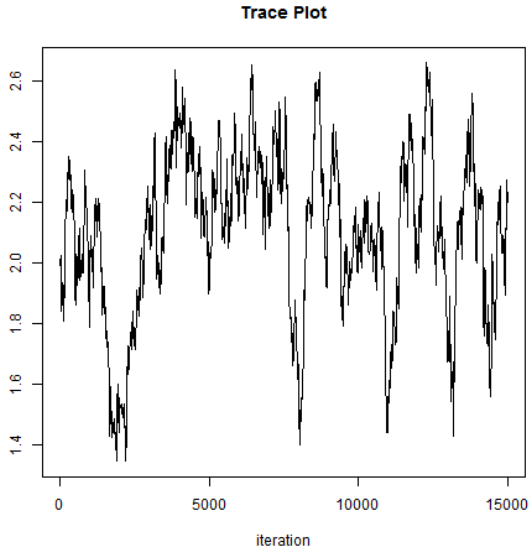
Bounded Adaption Metropolis



Standard Metropolis, $N(X_n, I_d)$



Standard Metropolis, $N(X_n, 0.01I_d)$



Standard Metropolis, $N(X_n, 0.001I_d)$

Figure 4.3: Trace plots (of coordinate 10) for the Pump Failure Model example for a Bounded Adaption Metropolis algorithm (top left), compared to Standard Metropolis algorithms with proposal distributions whose Gaussian covariance matrices are the d-dimensional identity (top right), 0.01 times this identity (bottom left), and 0.001 times this identity (bottom right).

Figure 4.3 shows the trace plots (of coordinate 10) for the pump failure model with

the BAM algorithm, and with the standard Metropolis algorithm with proposal kernels $N(X_n, I_d)$, $N(X_n, 0.01\, I_d)$, and $N(X_n, 0.001\, I_d)$.

For the non-adaptive algorithm with a fixed proposal kernel $N(X_n, I_d)$, not a single proposal was accepted for the whole 15000 iterations. This is a combination of a couple of factors. First, it is a rare event for 12 univariate normal proposal kernels to suggest all positive numbers when the variance for the each proposal kernel is 1 and the starting value, X_0, for each coordinate ranges from 0.6 to 2. Also, X_0 is the estimates from a previous MCMC run, so the evaluation of X_0 under the target density would be greater than most of the other points in the state space. Thus, accepting a new proposal, a point in the state space, over X_0 does not exactly have a high probability of happening unless the move is really small. The BAM algorithm overcomes this problem as it adjusts the proposal variance to suit for the target distribution of interest.

If we reduce down the scale of our proposal kernel for non-adaptive algorithms to $N(X_n, 0.01I_d)$ or $N(X_n, 0.001I_d)$, then the proposals are accepted more often. However, the mixing of the chains for these non-adaptive algorithms are still clearly not as good as for BAM. This indicates that our new Bounded Adaption Metropolis (BAM) adaptive MCMC algorithm performs better than standard Metropolis algorithms, even if their proposal scalings are adjusted manually to allow for reasonable acceptance rates.

It is our hope that the easily-verifiable ergodicity conditions presented herein will allow MCMC practitioners to make more widespread use of such adaptive MCMC algorithms, and thus benefit from their computational speed-ups without suffering from burdensome or uncheckable technical conditions.

Chapter 5

Adaptive Component-wise Multiple-Try Metropolis Sampling

5.1 Introduction

Markov chain Monte Carlo (MCMC) methods are widely used to analyze complex probability distributions, especially within the Bayesian inference paradigm. One of the most used MCMC algorithms is the Metropolis-Hastings (MH) algorithm, first developed by Metropolis et al. (Metropolis et al. 1953), and later expanded by Hastings (Hastings 1970). At each iteration the MH algorithm samples a candidate new state from a proposal distribution which is subsequently accepted or rejected. When the state space of the chain is high dimensional or irregularly shaped, finding a good proposal distribution that can be used to update *all* the components of the chain simultaneously is very challenging, often impossible. The optimality results for the acceptance rate of the Metropolis-Hastings algorithm (Gelman et al. 1996; Roberts and Rosenthal 2001) have inspired the development of the so-called *adaptive MCMC (AMCMC)* samplers that are designed to adapt their transition kernels based on the gradual information about the target that is collected through the very samples they produce.

Successful designs can be found in Haario et al. (2001), Haario et al. (2006), Turro et al. (2007),Roberts and Rosenthal (2009), Craiu et al. (2009), Giordani and Kohn (2010), and Vihola (2012) among others. Theoretical difficulties arise because the adaptive chains are no longer Markovian so the convergence must be proven on a case-by-case basis. Attempts at streamlining the theoretical validation process for AMCMC samplers have been increasingly successful including Atchadé and Rosenthal. (2005), Andrieu and Moulines (2006), Andrieu and Atchadé (2007), Roberts and Rosenthal (2007), Fort et al. (2011) and Craiu et al. (2015). For useful reviews of AMCMC we refer to Andrieu and Thoms (2008) and Roberts and Rosenthal (2009). Despite many success stories, it is our experience that existing adaptive strategies for MH may take a very long time to "learn" good simulation parameters or may be remain inefficient in high dimensions and with irregular shaped targets.

One can increase the computational efficiency if instead of using a full MH to update all the components at once, one chooses to update the components of the chain one-at-a-time. In this case the update rule follows the MH transition kernel but the acceptance or rejection is based on the target's conditional distribution of that component given all the other ones. More precisely, if we are interested in sampling from the continuous density $\pi(x) : \mathcal{X} \subset \mathbf{R}^d \to \mathbf{R}_+$; the component-wise MH (CMH) will update the ith component of the chain, x_i, using a proposal $y_i \sim T_i(\cdot|x_i)$ and setting the next value of the chain as

$$z = \begin{cases} (x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_d) & \text{w.p. } \alpha_i \\ x & \text{w.p. } 1 - \alpha_i \end{cases}$$

where

$$\alpha_i = \min\left\{1, \frac{T(x_i|y_i)\pi(y_i|x_{[-i]})}{T(y_i|x_i)\pi(x_i|x_{[-i]})}\right\},$$

and $\pi(\cdot|x_{[-i]})$ is the target conditional distribution of the ith component given all the other components $x_{[-i]} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$. One can see that the CMH replaces the

difficult problem of finding one good proposal in d dimensions with an apparently easier problem of finding d good 1-dimensional proposals. However, the latter task can also prove difficult if the conditional density $\pi(\cdot|x_{[-i]})$ turns out to have different properties as $x_{[-i]}$ varies. Intuitively, imagine that for a region of the sample space of $x_{[-i]}$ the proposal T_i should have a higher spread for the chain to mix well and for the remaining support T_i should have a smaller spread. Some success has been obtained in lower dimensions or for distributions with a well-known structure using the regional adaptive MCMC strategies of Craiu et al. (2009) or Bai et al. (2011a), but extending those approaches will be very cumbersome when d is even moderately large and the geography of π exhibits unknown irregularities. Other adaptive MCMC ideas proposed for the CMH too include Haario et al. (2005) where the authors propose to use component-wise random walk Metropolis (RWM) and to use the component-specific sample variance to tune the proposal's variance, along the same lines that were used by Haario et al. (2001) to adapt the proposal distribution for the joint RWM. Another intuitive approach is proposed in Roberts and Rosenthal (2009). The algorithm we propose here will be compared with these existing algorithms in the simulations section.

The strategy we propose here aims to close the gap that still exists between AMCMC and efficient CMH samplers. When contemplating the problem, one may be tempted to try to "learn" each conditional distribution $\pi(\cdot|x_{[-i]})$, but parametric models are likely not flexible enough and nonparametric models will face the curse of dimensionality even for moderate values of d. Note that here the difficult part is understanding how the conditional distribution changes as $x_{[-i]}$ varies, which is a $d-1$-dimensional problem.

Before getting to the technical description of the algorithm, we present here the intuitive idea behind our design. Within the CMH algorithm imagine that for each component we can propose m candidate moves, each generated from m different proposal distributions. Naturally, the latter will be selected to have a diverse range of variances so that we generate

some proposals close to the current location of the chain and some that are further away. If we assume that the transition kernel for each component is such that among the proposed states it will select the one that is most likely to lead to an acceptance, then one can reasonably infer that this approach will improve the mixing of the chain provided that the proposal distributions are reasonably calibrated. To mirror the discussion above, in a region where T_i should have small spread, one wants to have among the proposal distributions a majority with small variances, and similarly in regions where T_i should be spread out we want to include among our proposal densities a majority with larger variances.

This intuition can be put to rigorous test using an approach based on the Multiple-try Metropolis (MTM) that originated with Liu et al. (2000) and was further generalized by Casarin et al. (2013). The theoretical validity of the sampler is demonstrated.

Section 5.2 introduces a component-wise multiple-try Metropolis (CMTM) algorithm; in Section 5.3 we add the adaptive flavour to CMTM so the proposal distributions will modify according to the local shape of the target distribution and we prove the validity of our construction. Section 5.4 applies the adaptive CMTM algorithm to numerical examples. Section 5.5 compares the efficiency of the adaptive CMTM algorithm to other adaptive Metropolis algorithms.

5.2 Component-wise Multiple-Try Metropolis

5.2.1 Algorithm

Assume that a Markov chain $\{X_n\}$ is defined on $\mathcal{X} \subset \mathbf{R}^d$ with a target distribution π, and $T_1, \ldots, T_m$ are proposal distributions, each of which generates a new proposal y_j at every iteration. The transition rule for the MTM algorithm is described below.

1. Let $X_n = x$. Draw proposals $y_1, \ldots, y_m$ where $y_j \sim T_j(\cdot|x)$.

2. Compute

$$w_j(y_j, x) = \pi(y_j) T_j(x|y_j) \lambda_j(y_j, x), \tag{5.1}$$

for each y_j, where $\lambda_j(x, y)$ is a nonnegative symmetric function and $\lambda_j(x, y) > 0$ whenever $T_j(y|x) > 0$.

3. Select one $y = y_s$ out of $y_1, \ldots, y_m$ with probabilities proportional to $w_j(y_j, x)$.

4. Draw $x_1^*, \ldots, x_{s-1}^*, x_{s+1}^*, \ldots, x_m^*$ where $x_j^* \sim T_j(\cdot|y)$ and let $x_s^* = x$

5. Accept y with a probability

$$\rho = \min \left[1, \frac{w_1(y_1, x) + \ldots + w_m(y_m, x)}{w_1(x_1^*, y) + \ldots + w_m(x_m^*, y)}\right]$$

Convergence of the MTM to the target distribution is confirmed in Liu et al. (2000) and Casarin et al. (2013).

Throughout the chapter, we use the component-wise multiple-try Metropolis (CMTM) algorithm in which each coordinate is updated using an MTM transition kernel with Gaussian proposals for each coordinate. More precisely, suppose that we are updating the kth component of the current state x using m 1-dimensional proposals, $T_j(\cdot|x)$, $1 \leq j \leq m$. Then a candidate $y_j \sim T_j(\cdot|x)$ is generated by sampling $z_k \sim N(x_k, \sigma_{k,j}^2)$ and by replacing the k^{th} coordinate of current state x with z_k, i.e. $y_j = (x_1, \ldots, x_{k-1}, z_k, x_{k+1}, \ldots, x_d)$. Similarly, within the same iteration we get $x_j^* \sim T_j(\cdot|y)$, by replacing k^{th} coordinate of y with $z_k^* \sim N(y_k, \sigma_{k,j}^2)$.

Whether a proposal distribution is 'good' or not will depend on the current state of the Markov chain, especially if the target distribution π has irregular shaped support. In addition to choosing the m proposals, an added flexibility of the CMTM algorithm is that we can choose any nonnegative symmetric map λ with $\lambda_j(x, y) > 0$ whenever $T_j(y|x) > 0$.

In subsequent sections we show that the CMTM algorithm with a particular form of the function $\lambda(x, y)$ influences positively the mixing of the chain and its effects depend in a subtle manner on the local shape, i.e. the geography of the target distribution around the current state, and the proposal's scale.

Our choice of λ is guided by a simple and intuitive principle. Between two candidate moves y_1 and y_2 that are equally far from the current state we favour y_1 over y_2 if $\pi(y_1)$ is greater than $\pi(y_2)$, but if $\pi(y_1)$ is similar to $\pi(y_2)$, we would like CMTM to favour whatever candidate is further away from the current state. These simple rules lead us to consider

$$\lambda_j(x, y) = T_j(y|x)^{-1}\|(y - x)\|^{\alpha}, \tag{5.2}$$

where $\|\cdot\|$ is the Euclidean norm. Note that this choice of λ is possible because $T_j(y|x)$ is a symmetric function in x and y as it involves only one draw from a normal distribution with mean x_k.

Replacing (5.2) in the weights equation (5.1) results in

$$\begin{aligned} w_j(y_j, x) &= \pi(y_j)T_j(x|y_j)\lambda_j(y_j, x) \\ &= \pi(y_j)\|(y_j - x)\|^{\alpha}. \end{aligned} \tag{5.3}$$

With this choice of λ, the selection probabilities are only dependent on the value of the target density at the candidate point y_j and the size of the potential jump of the chain, were this candidate accepted. From (5.2) we can see that the size of α will balance of importance of the attempted jump distance from the current state over the importance of the candidate under π. However, while we understand the trade-off imposed by the choice of α for selecting a candidate move, it is less clear how it will impact the overall performance of the CMTM, e.g acceptance rate or average jump distance.

Therefore, it is paramount to gauge what are good choices for the parameter α for the mixing of the CMTM chain. In the next section we tackle this using the average squared jumping distance as the approximate measure of performance of a Markov chain. An estimate is obtained by averaging over the realized path of the chain. If a new proposal is rejected and $(X_{n+1} - X_n)^2$ is equal to zero, that contribution is not discarded. Because of this, the measure of efficiency takes into account not only the jump distance but also the acceptance rate, a combination that has turned out to be useful in other AMCMC designs (see for instance Craiu et al. 2009).

5.2.2 Optimal α

In order to study the influence of the parameter α on the CMTM efficiency we have conducted a number of simulation studies, some of which are described here.

We considered first a 2-dimensional mixture of two normal distributions

$$\frac{1}{2} * N(\mu_1, \Sigma_1) + \frac{1}{2} * N(\mu_2, \Sigma_2) \qquad (5.4)$$

where

$$\begin{cases} \mu_1 &= (5,0)^T \\ \mu_2 &= (15,0)^T \\ \Sigma_1 &= diag(6.25, 6.25) \\ \Sigma_2 &= diag(6.25, 0.25) \end{cases}$$

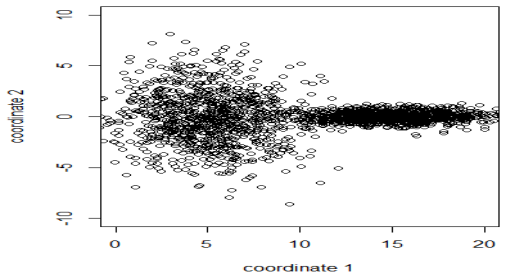


Figure 5.1: Target density plot. 2-dimensional mixture of two normals

An iid sample of size 2000 from (5.4) is plotted in Figure 5.1. We run the CMTM algorithm repeatedly with $\lambda_j(x, y_j)$ functions in (5.2) while changing the value of α from 0.1 to 15. We choose $m = 5$ as the number of proposals for each coordinate, while the proposal

variances $\sigma_{k,j}$'s are for each coordinate $1, 2, 4, 8$ and 16.

As we see in Figure 5.2, the proportion of each proposal distribution selected increases/decreases as α changes. As expected, when α increases we see the selection percentages of the proposal distributions with smaller $\sigma_{k,j}$'s drop and those with larger $\sigma_{k,j}$'s increase. Figure 5.2 shows, with larger α's, our algorithm favours proposal distributions with larger scales, which makes sense based on the equation (5.3).

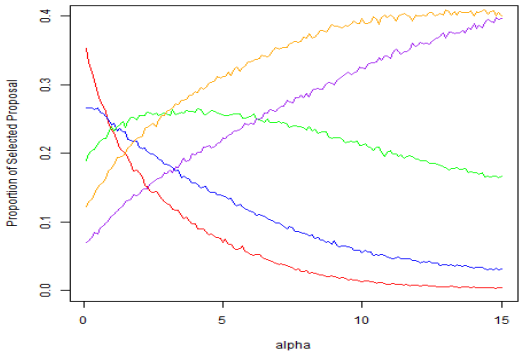
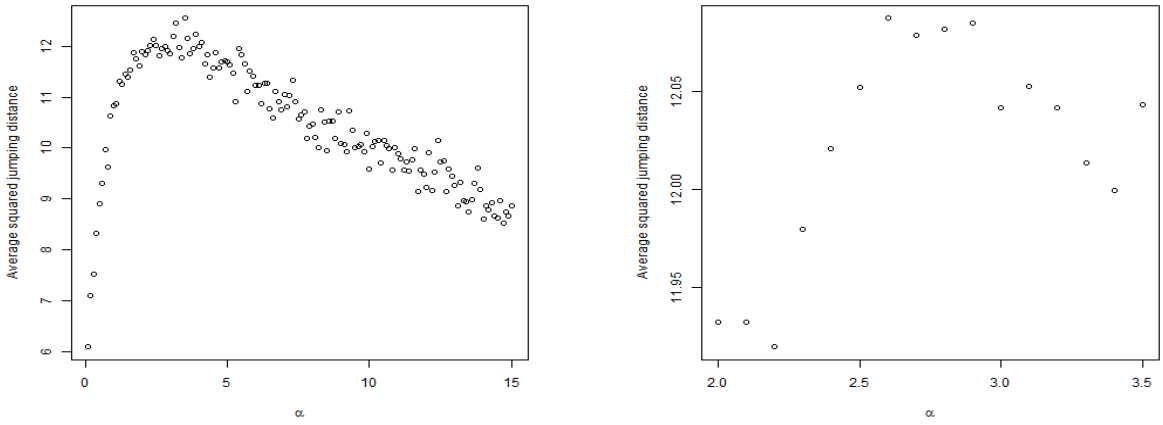


Figure 5.2: Proportion of proposal distribution selected. Coordinate 1: Red, Blue, Green, Orange and Purple lines show behaviour when $\sigma_{k,j} = 1, 2, 4, 8, 16$, respectively.

Figure 5.3 shows how the average squared jumping distance changes as the value of α changes. From Figure 5.3a, we see that the average squared jumping distance peaks in-between $\alpha = 2$ and $\alpha = 4$. Next, we run the CMTM algorithm independently 100 times for each of $\alpha = 2.0, 2.1, \ldots, 3.5$ and average the average squared jumping distances over the 100 runs. From Figure 5.3b we can infer that the highest efficiency is achieved for $\alpha \in (2.5, 3.2)$.



(a) (b) (averaged over 100 runs)

Figure 5.3: Two-Dimensional Mixture of two Gaussians: Mean squared jumping distance vs. α for one run (left panel) and averaged over 100 runs (right panel).

We also examined a 4-dimensional mixture of two normal distributions as our target density:

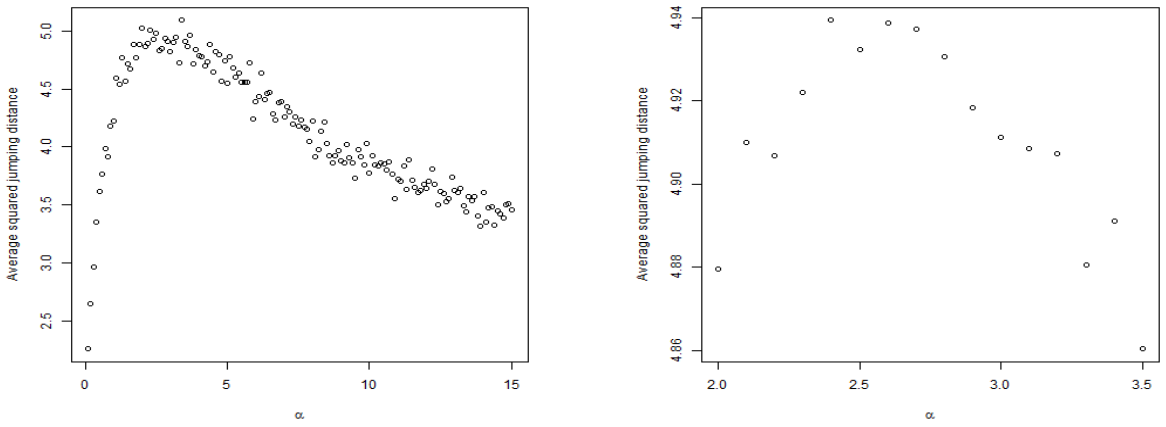
$$\frac{1}{2} * N(\mu_1, \Sigma_1) + \frac{1}{2} * N(\mu_2, \Sigma_2),$$

where

$$\begin{cases} \mu_1 & = (5, 5, 0, 0)^T \\ \mu_2 & = (15, 15, 0, 0)^T \\ \Sigma_1 & = diag(6.25, 6.25, 6.25, 0.01) \\ \Sigma_2 & = diag(6.25, 6.25, 0.25, 0.01). \end{cases}$$

The number of proposals, $m = 5$ and $\sigma_{k,j}$'s of the set of proposal distributions for each coordinate are $0.5, 1, 2, 4$ and 8. Figure 5.4 shows the results. Figure 5.4a shows the average squared jumping distance is the largest between $\alpha = 2$ and $\alpha = 4$. After 100 independent

replicates for each $\alpha = 2.0, 2.1, \ldots, 3.5$, we can see from Figure 5.4b that the average squared jumping distances are largest for $\alpha \in (2.5, 3.2)$.



(a) (b) (averaged over 100 runs)

Figure 5.4: 4-Dimensional Mixture of two Gaussians: Means squared jumping distance vs. α for one run (left panel) and averaged over 100 runs (right panel).

Other numerical experiments not reported here agree with the two examples presented and suggest that optimal values of α are between 2.5 and 3.2. Henceforth we fix $\alpha = 2.9$ in all simulations involving CMTM.

5.3 Adaptive Component-wise Multiple-Try Metropolis

5.3.1 CMTM Favours Locally 'Better' Proposal Distributions

The intuition behind our construction as described in the Introduction, relies on the idea that CMTM will automatically tend to choose the "right" proposal among the m possible ones. In this section we verify empirically that this is indeed the case.

We consider the same 2-dimensional mixture of normal distributions from Section 5.2.2 as our target distribution and run the CMTM algorithm. The simulation parameter are the same, $m = 5$ and $\sigma_{k,j} = 1, 2, 4, 8, 16$. As shown in Figure 5.1, the shape of our target distribution is quite different between the two regions of $\{X_{n,1} < 10\}$ and $\{X_{n,1} \geq 10\}$.

Table 5.1: Proportion of selected proposals

coordinate 1						coordinate 2					
	$\sigma_{1,j}$						$\sigma_{2,j}$				
	1	2	4	8	16		1	2	4	8	16
$X_{n,1} < 10$	0.05	0.15	0.28	0.31	0.22	$X_{n,1} < 10$	0.06	0.19	0.33	0.26	0.16
$X_{n,1} \geq 10$	0.04	0.12	0.25	0.34	0.25	$X_{n,1} \geq 10$	0.37	0.29	0.19	0.11	0.05

Table 5.2: Acceptance rate of selected proposals

coordinate 1						coordinate 2					
	$\sigma_{1,j}$						$\sigma_{2,j}$				
	1	2	4	8	16		1	2	4	8	16
$X_{n,1} < 10$	0.39	0.50	0.51	0.53	0.50	$X_{n,1} < 10$	0.46	0.52	0.50	0.47	0.44
$X_{n,1} \geq 10$	0.40	0.44	0.44	0.45	0.38	$X_{n,1} \geq 10$	0.44	0.41	0.30	0.28	0.28

Tables 5.1 and 5.2 present the proportion of candidate selection and acceptance rates for each proposal. We compare the proportion of proposals selected in the regions $\{X_{n,1} \geq 10\}$ and $\{X_{n,1} < 10\}$. While these regions are defined based on knowing the target exactly, they do not enter in any way in the design of the CMTM and are used here only to verify that the sampler indeed automatically adapts to local characteristics of the target. We can see that the CMTM favours the proposal distributions with smaller $\sigma_{k,j}$'s in the region $\{X_{n,1} \geq 10\}$ which seems appropriate given that in that region larger moves for the second coordinate will tend to be rejected. This pattern does not hold for the first coordinate for which larger moves are appropriate throughout the sample space. In addition, Table 5.1 shows that in the

region where $\{X_{n,1} < 10\}$ the proportions of proposal distribution selected are similar for the two coordinates. This is in line with what is expected since the target variances ($= 6.25$) are the same in both directions in that region.

This suggests that indeed the CMTM algorithm tends to choose the 'better' proposal distribution out of the available choices provided at each iteration. Also, as shown in Table 5.2, none of proposal distributions once selected has too high or too low acceptance rate even though their variances are quite different. This also hints that the CMTM assigns a selection probability to the multiple proposals at each iteration in such a way that improves the efficiency of the Markov chain by avoiding too low and too high acceptance rate. This observation will help us design the adaptive version of the CMTM.

5.3.2 Comparison with a Mixture Transition Kernel

An astute reader may wonder about a different strategy for using the different proposals that one may have at one's disposal. Maybe the most natural alternative is a random mixture of the component-wise Metropolis algorithms. The set of proposal distributions used in both algorithms is the same and we assign equal weights for the proposal distributions in the mixture. This comparison is to see empirically how automatically adjusting the selection probabilities of proposal distributions in the CMTM algorithm improves the efficiency of a MCMC algorithm compared to the uniform selection of proposal distributions. Our target distribution is the 4-dimensional mixture of two normals introduced in Section 5.2.2. The $\sigma_{k,j}$'s for both algorithms are $0.5, 1, 2, 4$ and 8 for each coordinate, and each independent run has 60000 iterations (15000 updates per coordinate) in total.

A useful comparison of the efficiency of MCMC algorithms is based on estimates of

integrated autocorrelation time (ACT) which can be calculated using

$$\kappa = 1 + 2\sum_{k=1}^{\infty} \rho_k,$$

where $\rho_k = Cov(X_0, X_k)/Var(X_0)$ is the autocorrelation at lag k. Higher ACT for a Markov chain implies successive samples are highly correlated, which reduces the effective information contained in any given number of samples produced by the chain. In this chapter, we calculate the ACT for each coordinate of a Markov chain and evaluate algorithms by comparing the worst (largest) coordinate-wise ACT.

Also, we use the effective sample size (ESS) to compare the efficiency of MCMC algorithms. Since ESS $= w/\kappa$, where w is the number of samples obtained from a Markov chain and κ is the ACT, one can see that ESS is tightly connected to the degree of correlation of the samples. One may intuitively interpret ESS the number of iid samples from the target that would contain the same amount of information about the target as the MCMC sample.

Table 5.3: CMTM. Proportion of proposal distribution selected

	$\sigma_{k,j}$				
	0.5	1	2	4	8
coord1	0.02	0.07	0.22	0.37	0.32
coord2	0.02	0.07	0.21	0.36	0.34
coord3	0.13	0.20	0.23	0.25	0.19
coord4	0.56	0.24	0.12	0.06	0.03

To compare the efficiency of two algorithms, we compare the ACT (and ESS) calculated from the runs of two algorithms. We also look at the CPU time taken for the runs (of 60000 iterations each) and the average squared jumping distances. We calculate the ACT only from the second-half the chain since we discard the first-half as burn-in, but we calculate the average squared jumping distance for the whole length of the chain.

Table 5.3 shows the proportion of each proposal distribution selected for the CMTM algorithm. It is quite different from what can be obtained by the uniform sampling, which

is used for the random mixture of the component-wise Metropolis algorithm. By assigning different selection probabilities than uniform, the CMTM algorithm controls the acceptance rate, avoiding too high and too low acceptance rate, as shown in Table 5.4.

Table 5.4: Acceptance rate on selected proposals

(a) Mix. of component-wise Metropolis

	$\sigma_{k,j}$				
	0.5	1	2	4	8
coord1	0.94	0.88	0.76	0.56	0.36
coord2	0.94	0.89	0.76	0.58	0.35
coord3	0.83	0.70	0.54	0.38	0.23
coord4	0.25	0.12	0.06	0.04	0.02

(b) CMTM

	$\sigma_{k,j}$				
	0.5	1	2	4	8
coord1	0.41	0.49	0.55	0.53	0.47
coord2	0.40	0.49	0.55	0.53	0.45
coord3	0.48	0.47	0.50	0.50	0.47
coord4	0.27	0.27	0.27	0.31	0.29

Table 5.5: Performance comparison (averaged over 100 runs)

(a) Mix. of component-wise Metropolis

	Min.	Median	Mean	Max.
cputime(s)	14.92	19.08	19.98	29.94
sq. jump	1.443	1.555	1.553	1.650
	coord1	coord2	coord3	coord4
ACT	1109.67	1105.99	39.93	68.74
ESS	27.04	27.13	751.31	436.45
ESS/cputime	1.35	1.36	37.60	21.84

(b) CMTM

	Min.	Median	Mean	Max.
cputime(s)	126.3	144.3	146.0	197.5
sq. jump	4.657	4.913	4.921	5.131
	coord1	coord2	coord3	coord4
ACT	316.79	318.28	9.68	13.59
ESS	94.70	94.26	3098.46	2207.06
ESS/CPUtime	0.65	0.65	21.22	15.11

Table 5.5 compares the performance of two algorithms. We see that the average squared jumping distance significantly improves with the CMTM compared to the random mixture of the component-wise Metropolis. We can also see that the ACT is smaller for the CMTM than the random mixture of the component-wise Metropolis. Thus, we conclude that a selection probability assignment by the CMTM algorithm through the $w_j(y_j|x)$ function in (5.3) improves the efficiency of the CMTM algorithm. One concern for the CMTM is the

increased computation cost. We see in Table 5.5 the runtime for the CMTM is about 7 to 8 times longer than the random mixture of the component-wise Metropolis. This results in lower ESS/CPUtime for the CMTM compared to the random mixture of the component-wise Metropolis (Neal 2011). We will discuss the efficiency issue further in Section 5.5.

5.3.3 The Adaptive CMTM

Given its propensity to choose the best candidate put forward by the proposal distributions, it is reasonable to infer that CMTM's performance will be roughly aligned with the most suitable proposal for the region where the chain current state lies. The other side of the coin is that a whole set of bad proposals will compromise the efficiency of the CMTM algorithm. Therefore, we focus our efforts in developing an adaptive CMTM (AMCTM) design that aims to minimize, possibly annihilate, the chance of having at our disposal only poorly calibrated proposal distributions in any region of the space.

The adaption strategy is centred on finding well-calibrated values for the set $S_k = \{\sigma_{k,j} : 1 \leq j \leq m\}$ for every coordinate $1 \leq k \leq d$. Note that S_k varies across coordinates.

Consider an arbitrarily fixed coordinate and suppose we label the m proposal distributions such that $\sigma_{k,1} < \sigma_{k,2} < \ldots < \sigma_{k,m}$. Changes in the kernel occur at fixed points in the simulation process, called *adaption points*. The changes will occur only if an alarm is triggered at an adaption point. An alarm is triggered only if we notice that the candidates generated by the proposal distributions with the smallest scale $\sigma_{k,1}$ or the largest one $\sigma_{k,m}$ are over-selected. For instance, suppose that in an inter-adaption time interval the candidates generated by $\sigma_{k,1}$ are selected more than 40% of the time. The latter threshold is more than double the selection percentage for the least selected candidate since, if we denote p_j the frequency of selecting the candidate generated using $\sigma_{k,j}$ we have $\sum_{p_j} = 1 \geq m \min p_j$ and $m = 5$ in our implementation. The high selection percentage for $\sigma_{k,1}$ suggests that the

chain tends to favour, when updating the kth coordinate, proposals with smaller scale so the ACMTM design requires to: 1) halve the value of $\sigma_{k,1}$; 2) not modify the largest element in S_k; 3) recalculate the intermediate values, $\sigma_{k,2}, \ldots, \sigma_{k,m-1}$ to be equidistant between $\sigma_{k,1}$ and $\sigma_{k,m}$ on the log-scale.

Similarly, if the largest element in S_k, $\sigma_{k,m}$, produces proposals that are selected with frequency higher than 40% we consider this indicative of the chain requiring larger scale proposals for the kth coordinate and we replace $\sigma_{k,m}$ by its double, we keep the smallest value in S_k and the intermediate values are recalculated to be equidistant on log-scale. Notice that the range of the $\sigma_{k,j}$'s never decreases in our adaption strategy.

If neither the smallest nor the largest elements in S_k produce proposals that are selected more than 40% of the time, we wait until the algorithm reaches the next 'adaption point' and recalculate the proportion of each proposal candidate being selected during the last inter-adaption time interval. After recalculating the proportions, we again see if an alarm is triggered by over-selection (more than 40%) of either the smallest scale $\sigma_{k,1}$ or the largest one $\sigma_{k,m}$.

There is one minor technical detail required in our adaptive CMTM algorithm to ensure the convergence of the algorithm. The convergence of our adaptive CMTM algorithm is proved in Section 5.3.5.

- Fix a (large) constant $L > 0$ and a (really small) constant $\epsilon > 0$. Let $\sigma_{n,k,j}$ be the $\sigma_{k,j}$ used at n-th iteration in our adaptive CMTM algorithm. If $\sigma_{n,k,j}$ obtained from the adaption is greater than L, set $\sigma_{n,k,j} = L$. If $\sigma_{n,k,j}$ obtained from the adaption is less ϵ, set $\sigma_{n,k,j} = \epsilon$. (Of course the initial $\sigma_{n,k,j}$, $\sigma_{0,k,j}$, should be $\epsilon \leq \sigma_{0,k,j} \leq L$.)

5.3.4 To Adapt or Not To Adapt?

We compare the ACMTM algorithm with the CMTM algorithm without adaption to see if the adaption indeed improves the efficiency of the algorithm. We use the 4-dimensional mixture of two normal distributions from Section 5.2.2 as our target distribution. Both algorithms are run for 40000 iterations (10000 updates per coordinate). The $\sigma_{k,j}$'s for the non-adaptive algorithm are given in Table 5.6a and they are the same as the starting $\sigma_{k,j}$'s for the adaptive algorithm. We also provide in Table 5.6b the final versions of the $\sigma_{k,j}$'s obtained after the last adaption in one random run of ACMTM. For this particular run, the last adaption occurred right after 3600 iterations out of 40000 iterations in total. The comparison is based on 100 independent replicates, each of which yields the average squared jumping distance and the ACT.

Performance comparison is shown in Table 5.7.

Table 5.6: Ending $\sigma_{k,j}$'s

(a) Non-adaptive CMTM

	coord1	coord2	coord3	coord4
prop1	16	16	16	1
prop2	32	32	32	2
prop3	64	64	64	4
prop4	128	128	128	8
prop5	256	256	256	16

(b) Adaptive CMTM

	coord1	coord2	coord3	coord4
prop1	2.0000	2.0000	0.2500	0.0625
prop2	6.7272	6.7272	1.4142	0.2500
prop3	22.6274	22.6274	8.0000	1.0000
prop4	76.1093	76.1093	45.2548	4.0000
prop5	256.0000	256.0000	256.0000	16.0000

Table 5.7: Performance comparisons (averaged over 100 runs)

(a) Non-adaptive CMTM

	Min.	Median	Mean	Max.
cputime(s)	79.21	90.98	92.88	114.60
sq. jump	3.463	3.688	3.683	4.036
	coord1	coord2	coord3	coord4
ACT	336.82	332.02	19.39	26.00
ESS	59.38	60.24	1031.60	769.10
ESS/CPUtime	0.64	0.65	11.11	8.28

(b) Adaptive CMTM

	Min.	Median	Mean	Max.
cputime(s)	77.85	92.49	92.61	120.10
sq. jump	4.180	4.715	4.693	5.181
	coord1	coord2	coord3	coord4
ACT	249.08	249.96	13.56	11.47
ESS	80.30	80.01	1474.81	1743.56
ESS/CPUtime	0.87	0.86	15.93	18.83

Table 5.8: Proportion of proposal distribution selected

(a) Non-adaptive CMTM

	coord1	coord2	coord3	coord4
prop1	0.56	0.55	0.56	0.57
prop2	0.24	0.25	0.24	0.24
prop3	0.11	0.12	0.11	0.11
prop4	0.06	0.05	0.06	0.05
prop5	0.03	0.03	0.03	0.03

(b) Adaptive CMTM

	coord1	coord2	coord3	coord4
prop1	0.33	0.33	0.17	0.31
prop2	0.46	0.45	0.44	0.50
prop3	0.16	0.16	0.32	0.14
prop4	0.05	0.05	0.06	0.04
prop 5	0.01	0.01	0.01	0.01

Table 5.9: Acceptance rate on selected proposals

(a) Non-adaptive CMTM

	coord1	coord2	coord3	coord4
prop1	0.23	0.24	0.16	0.17
prop2	0.23	0.23	0.17	0.16
prop3	0.24	0.22	0.15	0.18
prop4	0.27	0.24	0.16	0.18
prop5	0.24	0.21	0.17	0.22

(b) Adaptive CMTM

	coord1	coord2	coord3	coord4
prop1	0.49	0.47	0.43	0.45
prop2	0.41	0.41	0.44	0.42
prop3	0.37	0.39	0.41	0.38
prop4	0.34	0.40	0.35	0.37
prop5	0.33	0.34	0.39	0.46

We notice that 'prop1' was the most favoured proposal distribution for every coordinate

if there was no adaption (Table 5.8a) whereas with adaption it was 'prop2' selected most often (Table 5.8b). The $\sigma_{k,j}$'s got smaller with adaption (Table 5.6), and the two smallest $\sigma_{k,j}$'s obtained only after the multiple adaptions are the most favoured ones in the adaptive run (Table 5.8b). In return, the acceptance rates in the adaptive algorithm increase to the range of 0.36 to 0.50 compared to 0.15 to 0.27 in the non-adaptive algorithm, as shown in Table 5.9. It is important to note that the increase in acceptance rates is coupled with an increase in average square distance, as shown in Table 5.7. Altogether we can see from the same Table that these result in important reductions for the ACT (thus increase for the ESS and ESS/CPUtime). The runtime between the non-adaptive and the adaptive algorithm is pretty much the same as seen in Table 5.7. Altogether, the results show that using an adaptive strategy made a significant difference.

5.3.5 Convergence of Adaptive CMTM

Theorem 12. *Consider the adaptive CMTM algorithm in Section 5.3.3 to sample from state space $\mathcal{X}$ that is an open subset of $\mathbf{R}^d$ for some $d \in \mathbf{N}$. Let π be a target probability distribution, which has a continuous positive density on $\mathcal{X}$ with respect to the Lebesgue measure. Then, the adaptive CMTM algorithm converges to stationarity as in*

$$\lim_{n\to\infty} \sup_{A\in\mathcal{F}} |\mathbf{P}(X_n \in A) - \pi(A)| = 0. \tag{5.5}$$

Proof. In our adaptive algorithm, the range of the $\sigma_{k,j}$'s never decreases. Also, it is assumed in Section 5.3.3 that $\sigma_{n,k,j}$'s obtained from the adaptions are bounded above by a (large) constant $L > 0$ and bounded below by a (really small) constant $\epsilon > 0$. Thus, adaptions make the range of the $\sigma_{k,j}$'s increase and if the algorithm hits the bound for all the $\sigma_{n,k,j}$'s it will stop adapting. Therefore, $P(\tau < \infty) = 1$, where τ is some stopping time for adaption, or in other words, there is no adaption after time τ.

Then by Proposition 3 of Roberts and Rosenthal (2007), the adaptive CMTM algorithm converges to π as in (5.5). □

5.4 Applications

In the following examples we compare the CMTM and the ACMTM when started with the same set of $\sigma_{k,j}$'s. Note that CPU times for the CMTM and the ACMTM are comparable when both run for the same number of iterations. Thus, we don't calculate ESS/CPUtime when comparing two algorithms.

In a second comparison we also consider the CMTM and the CMH samplers in which we use the $\sigma_{k,j}$'s identified by the ACMTM adaptive process in the following way: we run one ACMTM sampler started at $\sigma_{k,j} = 2^j$ for $1 \leq j \leq m$ and $1 \leq k \leq d$ and we use the final $\sigma_{k,j}$'s from the run to set up the CMTM and the m CMH samplers. If we assume the labelling such that for each coordinate k we have $\sigma_{k,1} \leq \ldots \leq \sigma_{k,m}$ then the jth CMH sampler uses $\sigma_{k,j}$ to generate moves in the kth coordinate, $1 \leq k \leq d$. For the CMTM we use all the $\sigma_{k,j}$'s identified by the adaptive process. Throughout this section we use $m = 5$.

Evey independent run in this section has $5000d$ iterations (5000 updates per coordinate) in total. Only the second half of the chain's run is used to calculate the ACT. We average the ACT over 50 or 100 independent runs, and with that average we calculate ESS.

5.4.1 Variance Components Model

The Variance Components Model (VCM) is a typical hierarchical model, well-used in Bayesian statistics community. Here, we use the data on batch to batch variation in dyestuff yields. The data were introduced in Davies (1967) and later analyzed by Box and Tiao (1973). The Bayesian set-up of the Variance Components Model on dyestuff yields is also well-described in Roberts and Rosenthal (2004). The data records yields on dyestuff of 5 samples, from

each of 6 randomly chosen batches. The data is shown in Table 5.10.

Table 5.10: Dyestuff Batch Yield (in grams)

Batch 1	1545	1440	1440	1520	1580
Batch 2	1540	1555	1490	1560	1495
Batch 3	1595	1550	1605	1510	1560
Batch 4	1445	1440	1595	1465	1545
Batch 5	1595	1630	1515	1635	1625
Batch 6	1520	1455	1450	1480	1445

Let y_{ij} be the yield on the dyestuff batch, with i indicating which batch it is from and j indexing each individual sample from the batch. The Bayesian model is then constructed as:

$$y_{ij}|\theta_i, \sigma_e^2 \sim N(\theta_i, \sigma_e^2), \qquad i = 1, 2, ..., K, \quad j = 1, 2, ..., J$$

where $\theta_i|\mu, \sigma_\theta^2 \sim N(\mu, \sigma_\theta^2)$. θ_i's are conditionally independent of each other given μ, σ_θ^2. The priors for the $\sigma_\theta^2, \sigma_e^2$ and μ are: $\sigma_\theta^2 \sim IG(a_1, b_1)$, $\sigma_e^2 \sim IG(a_2, b_2)$ and $\mu \sim N(\mu_0, \sigma_0^2)$. Thus, the posterior density function of this VCM model is

$$f(\sigma_\theta^2, \sigma_e^2, \mu, \theta_i|y_{ij}, a_1, a_2, b_1, b_2, \sigma_0^2) \propto$$
$$(\sigma_\theta^2)^{-(a_1+1)} e^{-b_1/\sigma_\theta^2} (\sigma_e^2)^{-(a_2+1)} e^{-b_2/\sigma_e^2} e^{-(\mu-\mu_0)^2/2\sigma_0^2} \prod_{i=1}^{K} \frac{e^{(\theta_i-\mu)^2/2\sigma_\theta^2}}{\sigma_\theta} \prod_{i=1}^{K}\prod_{j=1}^{J} \frac{e^{(y_{ij}-\theta_i)^2/2\sigma_e^2}}{\sigma_e}$$

We set the hyperparameters $a_1 = a_2 = 300$ and $b_1 = b_2 = 1000$, making inverse gamma priors very concentrated. We also set $\sigma_0^2 = 10^{10}$.

Figure 5.5 shows ESS of the CMTM algorithms with and without adaption. For both CMTM algorithms (with and without adaption), the starting $\sigma_{k,j}$'s are 0.1, 0.2, 0.4, 0.8 and 1.6 for every coordinate.

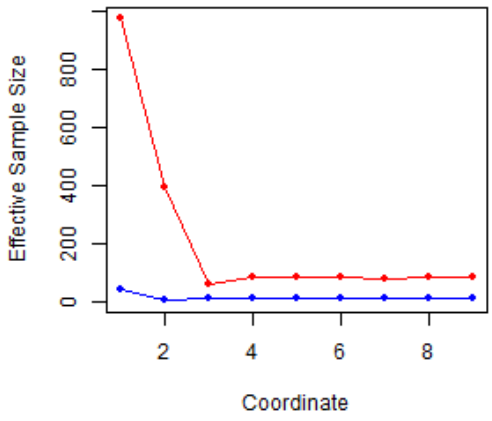


Figure 5.5: Adaptive CMTM vs. non-adaptive CMTM. Variance components model. The red represents the adaptive CMTM runs and the blue represents the non-adaptive CMTM runs. ESS is calculated after averaging ACT over 50 independently replicated runs.

Note that ACMTM yields ESS's that are considerably larger than the corresponding values for CMTM.

Next, we want to compare the standard (non-adaptive) CMTM algorithm with the standard CMH.

The $\sigma_{k,j}$'s used to set up the CMH and CMTM samplers can be found in Table 5.11. With this proposal scales, we run the standard CMTM and the standard component-wise Metropolis and compare the ESS from these two algorithms, averaged over 100 runs.

Table 5.11: $\sigma_{k,j}$'s used in comparing the standard CMTM and the standard component-wise Metropolis. Variance components model

	coord1	coord2	coord3	coord4	coord5	coord6	coord7	coord8	coord9
prop1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
prop2	3.36	4.76	2.83	2.38	3.36	3.36	3.36	3.36	3.36
prop3	11.31	22.63	8.00	5.66	11.31	11.31	11.31	11.31	11.31
prop4	38.05	107.63	22.63	13.45	38.05	38.05	38.05	38.05	38.05
prop5	128.00	512.00	64.00	32.00	128.00	128.00	128.00	128.00	128.00

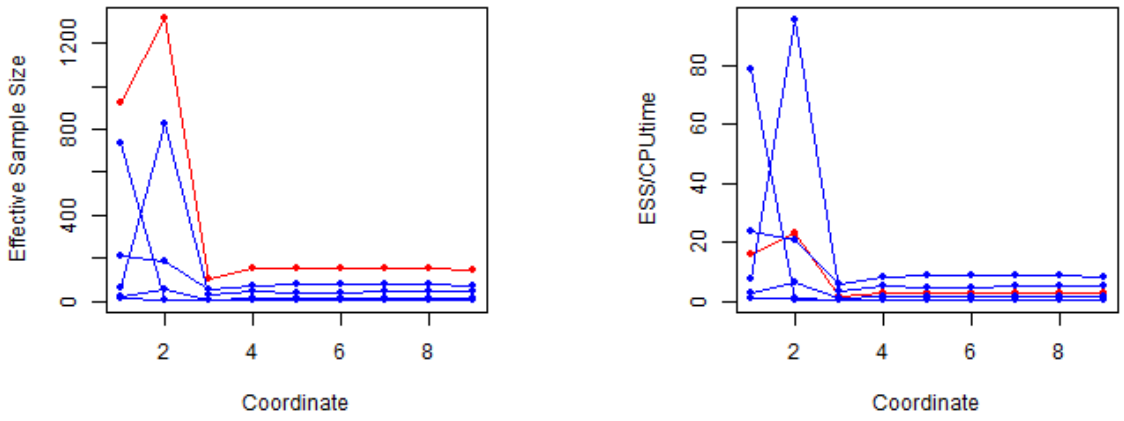


Figure 5.6: Standard (non-adaptive) CMTM vs. standard component-wise Metropolis. Variance components model. The red represents the CMTM runs and the blue represents the component-wise Metropolis runs. ESS is calculated after averaging ACT over 100 replicative runs.

Figure 5.6 shows that if we ignore the CPUtime loss for the CMTM algorithm compared to the component-wise Metropolis algorithm, the CMTM algorithm is the most efficient one based on ESS. However, when we divide ESS by CPUtime to account for the computation time, the CMTM becomes the third best algorithm compared with the 5 different component-wise Metropolis algorithms. We will discuss about this efficiency issue further in Section 5.5.

5.4.2 "Banana-shaped" Distribution

The "Banana-shaped" distribution was originally presented in Haario et al. (1999) as an irregularly-shaped target that may call for different proposal distributions for the different parts of the state space.

The target density function of the "banana-shaped" distribution is constructed as $f_B = f \circ \phi_B$, where f is the density of $d-$dimensional multivariate normal distribution $N(\mathbf{0}, \text{diag}(100, 1, 1, \ldots, 1))$ and $\phi_B(\mathbf{x}) = (x_1, x_2 + Bx_1^2 - 100B, x_3, \ldots, x_d)$. $B > 0$ is the nonlinearity pa-

rameter and the non-linearity or "bananacity" of the target distribution increases with B. The target density function is

$$f_B(x_1, x_2, \ldots, x_d) \propto exp[-x_1^2/200 - \frac{1}{2}(x_2 + Bx_1^2 - 100B)^2 - \frac{1}{2}(x_3^2 + x_4^2 + \ldots + x_d^2)].$$

We set $B = 0.01$ and $d = 10$ and set the starting $\sigma_{k,j}$'s equal to $0.1, 0.2, 0.4, 0.8$ and 1.6 for every coordinate. The results are shown in Figure 5.7. You see that the ESS is larger in every coordinate for the adaptive CMTM algorithm compared to the non-adaptive CMTM algorithm, confirming that the adaption improved the efficiency in this example.

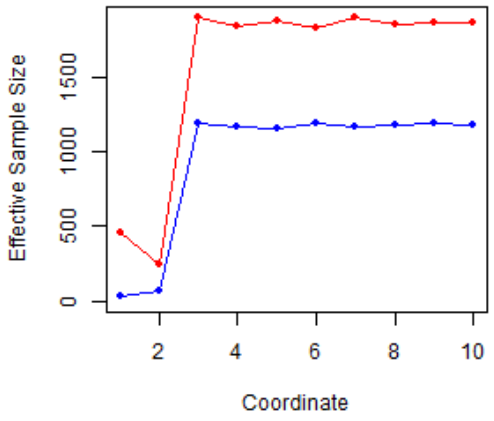


Figure 5.7: Adaptive CMTM vs. non-adaptive CMTM. "Banana-shaped" distribution. The red represents the adaptive CMTM runs and the blue represents the non-adaptive CMTM runs. ESS is calculated after averaging ACT over 50 replicative runs.

In the second comparison we notice that in 45 out of 50 independently replicated runs, the starting $\sigma_{k,j}$'s did not change under the adaptive CMTM algorithm. Therefore, we take 1,2,4,8,16 for every coordinate as our default $\sigma_{k,j}$'s for the standard CMTM algorithm we run here. And for the standard CMH algorithms, we run with five different sets of proposal stan-

dard deviation, $(1, 1, \cdots, 1)$, $(2, 2, \cdots, 2)$, $(4, 4, \cdots, 4)$, $(8, 8, \cdots, 8)$, and $(16, 16, \cdots, 16)$, with one set at a time. As we see in Figure 5.8, ESS is larger in every coordinate for the CMTM compared to the component-wise Metropolis if CPUtime is ignored. If we look at ESS/CPUtime, the CMTM is in the middle of the pack in terms of efficiency.

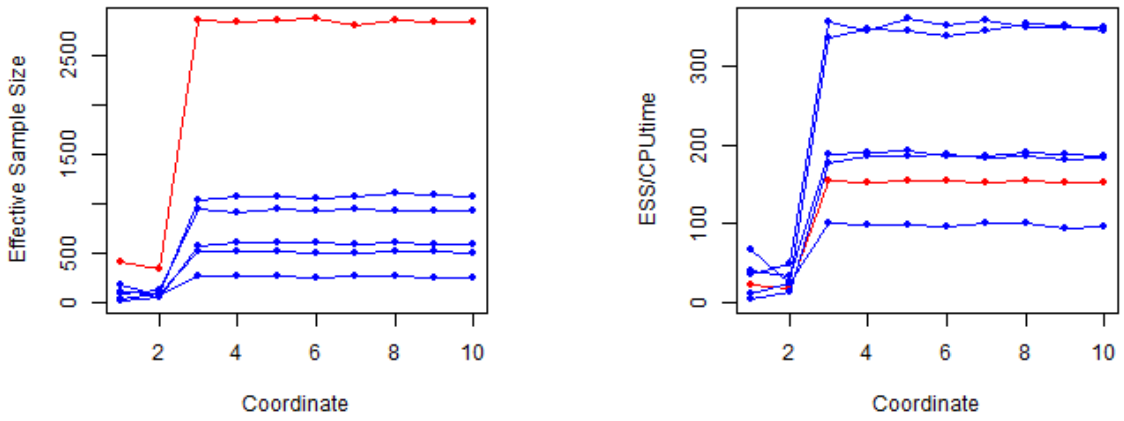


Figure 5.8: Standard (non-adaptive) CMTM vs. standard component-wise Metropolis. "Banana-shaped" distribution. The red represents the CMTM runs and the blue represents the component-wise Metropolis runs. ESS is calculated after averaging ACT over 50 replicative runs.

5.4.3 Orange Tree Growth Data

The orange tree growth data was first presented by Draper and Smith (1981) and was further analyzed by Lindstrom and Bates (1990). Here, we follow the Bayesian model set-up specified in Craiu and Lemieux (2007). The data consists of the measures of trunk circumferences of five different orange trees on seven different time points. You can find the data in Table 5.12.

Table 5.12: Growth of Orange Trees

Age (days)	Circumference (mm)				
	Tree1	Tree2	Tree3	Tree4	Tree5
118	30	33	30	32	30
484	58	69	51	62	49
664	87	111	75	112	81
1004	115	156	108	167	125
1231	120	172	115	179	142
1372	142	203	139	209	174
1582	145	203	140	214	177

Let y_{ij} be the trunk circumference measure, where $i = 1, \ldots, 5$ indexes for the five different trees and $j = 1, \ldots, 7$ indexes for the seven different time points. The logistic growth model has $y_{ij} \sim N(\mu_{ij}, \sigma_c^2)$ where

$$\mu_{ij} = \frac{exp(\theta_{i1})}{1 + (exp(\theta_{i2}) - 1)exp(-exp(\theta_{i3})x_j)}$$

with x_j being the time point. The priors for the σ_c^2 and the θ_{ik}, $k = 1, 2, 3$, are: $\sigma_c^2 \sim IG(0.001, 0.001)$ and $\theta_{ik} \sim N(0, 100)$. Note that θ_{ik}'s are independent of each other and y_{ij} are conditionally independent of each other given μ_{ij} and σ_c^2. Thus, the posterior density of interest is

$$f(\theta_{ik}, \sigma_c^2 | y_{ij}, x_j) \propto (\sigma_c^2)^{-(0.001+1)} e^{-0.001/\sigma_c^2} \prod_{i=1}^{5} \prod_{k=1}^{3} \frac{e^{(\theta_{ik})^2/200}}{10} \times \prod_{i=1}^{5} \prod_{j=1}^{7} \frac{e^{(y_{ij} - \frac{exp(\theta_{i1})}{1+(exp(\theta_{i2})-1)exp(-exp(\theta_{i3})x_j)})^2/2\sigma_c^2}}{\sigma_c}.$$

First, we compare the adaptive CMTM with the non-adaptive CMTM. The starting $\sigma_{k,j}$'s here for the both algorithms are 1, 2, 4, 8, and 16 for every coordinate and the results, shown

in Figure 5.9, confirm that the ESS is larger in every coordinate for the adaptive CMTM algorithm.

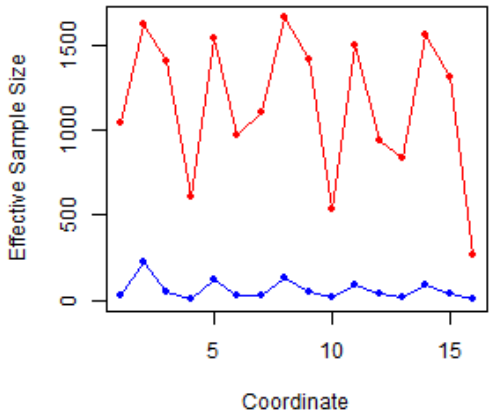


Figure 5.9: Adaptive CMTM vs. non-adaptive CMTM. Orange tree growth data. The red represents the adaptive CMTM runs and the blue represents the non-adaptive CMTM runs. ESS is calculated after averaging ACT over 50 replicative runs.

When comparing the CMTM with the CMH the proposal scales used can be found in Table 5.13

Table 5.13: $\sigma_{k,j}$'s used in comparing the standard CMTM and the standard component-wise Metropolis. Orange tree growth data

	coord1	coord2	coord3	coord4	coord5	coord6	coord7	coord8
prop1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
prop2	2.38	2.83	2.83	2.38	2.83	2.38	2.38	2.83
prop3	5.66	8.00	8.00	5.66	8.00	5.66	5.66	8.00
prop4	13.45	22.63	22.63	13.45	22.63	13.45	13.45	22.63
prop5	32.00	64.00	64.00	32.00	64.00	32.00	32.00	64.00
	coord9	coord10	coord11	coord12	coord13	coord14	coord15	coord16
prop1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
prop2	2.38	2.38	2.83	2.38	2.38	2.83	2.38	22.63
prop3	5.66	5.66	8.00	5.66	5.66	8.00	5.66	512.00
prop4	13.45	13.45	22.63	13.45	13.45	22.63	13.45	11585.24
prop5	32.00	32.00	64.00	32.00	32.00	64.00	32.00	262144.00

The results for the runs are shown in Figure 5.10. Again, ESS is larger in every coordinate for the CMTM algorithm compared to the component-wise Metropolis algorithm, but ESS/CPUtime is only the fourth largest for the CMTM algorithm.

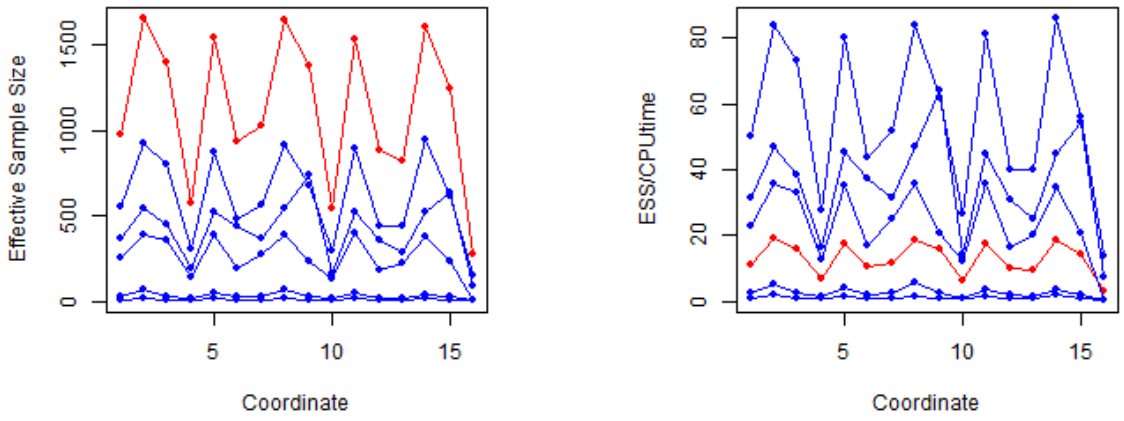


Figure 5.10: Standard (non-adaptive) CMTM vs. standard component-wise Metropolis. Orange tree growth data. The red represents the CMTM runs and the blue represents the component-wise Metropolis runs. ESS is calculated after averaging ACT over 100 replicative runs

5.5 Comparsion of Adaptive Algorithms

The CMTM algorithm has a higher computational cost compared to the standard component-wise Metropolis algorithm. For the same number of iterations, CMTM takes longer CPU time as it has to evaluate $2m - 1$ points under the target distribution at each iteration compared to 1 for the simple Metropolis algorithm. (m is the number of proposals for each iteration.) On the other hand, CMTM does improve efficiency per iteration as implied by the increase in ESS in Table 5.6, Table 5.8, and Table 5.10. In Section 5.4, the magnitude of ESS difference between the CMTM and the standard component-wise Metropolis, or equivalently the magnitude of efficiency gain, was dependent on the proposal variances of the standard component-wise Metropolis. Moreover, finding the best or close to the best proposal scales for the component-wise Metropolis is not always easy. If we fail to find a 'good' scale for the component-wise Metropolis or even the full-dimensional Metropolis these samplers will do a

lot worse than the CMTM algorithm, even after accounting for the additional computation cost with the CMTM. In a practical application one cannot know which one of the CMH's designed along the lines used in the previous section is better than the ACMTM until all the CMH's have been run and compared. Even then there is no guarantee that the most efficient CMH will be surpass the ACMTM. In our applications we can see that the ACMTM represents a solid bet in terms of efficiency even after accounting for CPU time.

Researchers have tried several adaptive Metropolis algorithms to find a 'good' scale for the Metropolis algorithm. We compare a few known adaptive Metropolis algorithms with the ACMTM algorithm.

The adaptive Metropolis algorithm we examine are: Adaptive Metropolis-within-Gibbs (AMwG) algorithm from Roberts and Rosenthal (2009), Single Component Adaptive Metropolis (SCAM) algorithm from Haario et al. (2005), and Adaptive Metropolis (AM) algorithm from Haario et al. (2001). The AMwG algorithm uses the finding that the optimal acceptance rate for one-dimensional Metropolis algorithm is 0.44 (Gelman et al. (1996), Roberts and Rosenthal (2001)) and adjusts the proposal variance to get the acceptance rate close to 0.44 for each coordinate. The SCAM algorithm finds the empirical variance for each coordinate from the entire history of the Markov chain and tunes the proposal variance for each coordinate based on this. The AM algorithm also looks at the entire history of the chain and tunes the proposal's full-dimensional covariance matrix using the empirical covariance matrix calculated from the samples collected up to that iteration.

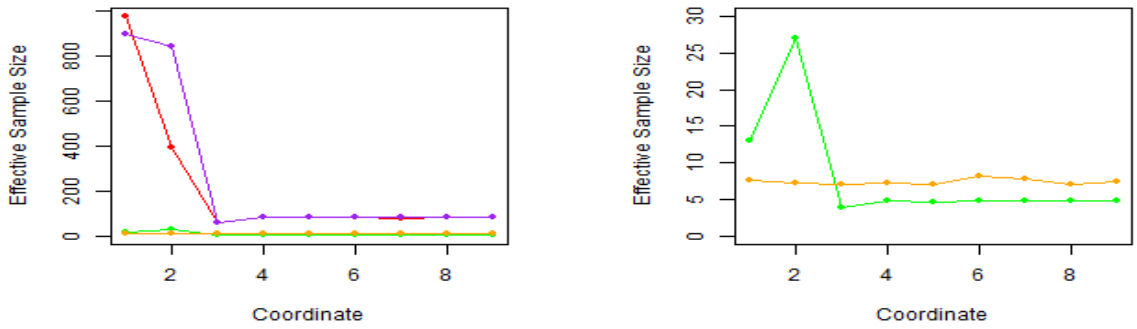
We look at the three examples from Section 5.4 and compare how these adaptive algorithms fare against the ACMTM algorithm. For the latter the default $\sigma_{k,j}$'s are 0.1, 0.2, 0.4, 0.8 and 1.6 for every coordinate.

For the AMwG algorithm, the default σ_j is 1 for each coordinate, and it gets adjusted upward if the acceptance rate is higher than 0.44 based on the recent "batch" of 100 iterations and adjusted downward if the acceptance rate is lower than 0.44. The σ_j is adjusted by adding

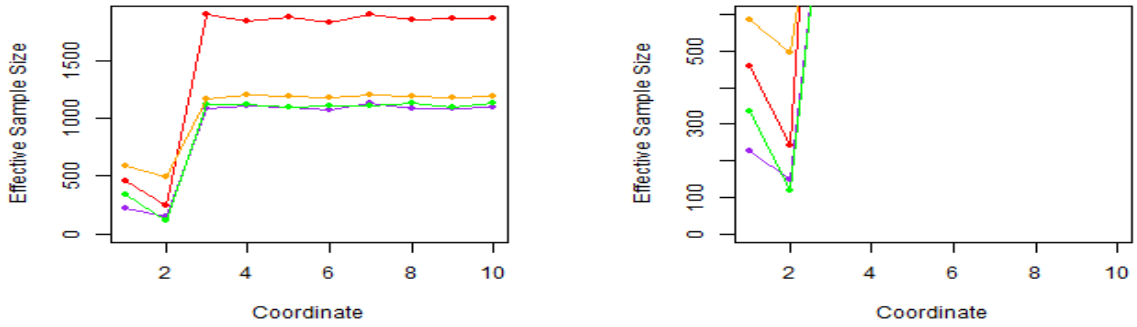
or subtracting $\min(0.05, \sqrt{(h)})$ in log scale based on the h-th "batch" of 100 iterations. One can see how this strategy may not be ideal when the chain alternates between regions in which the proposal must have different scales.

For the SCAM algorithm, the default σ_j for each coordinate is 1, and the chain runs for $10d$ iterations (10 updates per coordinate) with the default σ_j's. After $10d$ iterations, the proposal variance is tuned based on the empirical variance for each coordinate from the entire history of the chain, multiplied by the scaling parameter 2.38^2 given by Gelman et al. (1996).

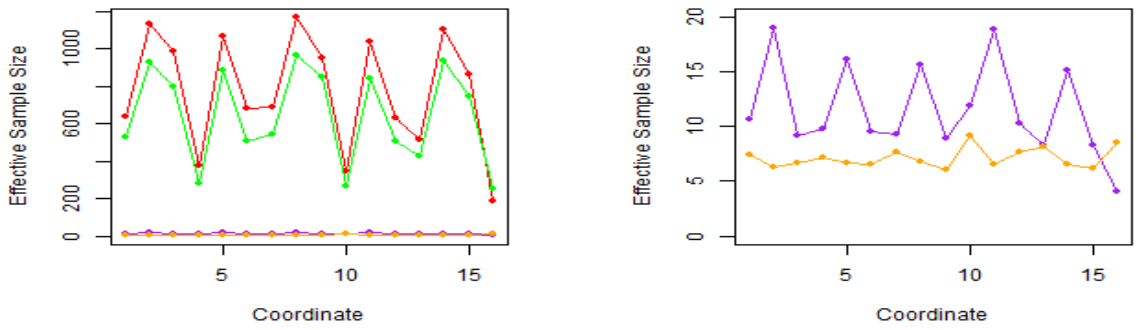
For the AM, the default Σ is an identity matrix, and after 100 iterations with default Σ, the algorithm starts to tune the proposal covariance matrix by multiplying the empirical covariance matrix with the choice of scaling parameter $2.38^2/d$ (d is the dimension of the Markov chain.) given by Gelman et al. (1996). The results of the comparison of these four adaptive algorithms are shown in Figure 5.11 and Figure 5.12.



(a) Variance components model

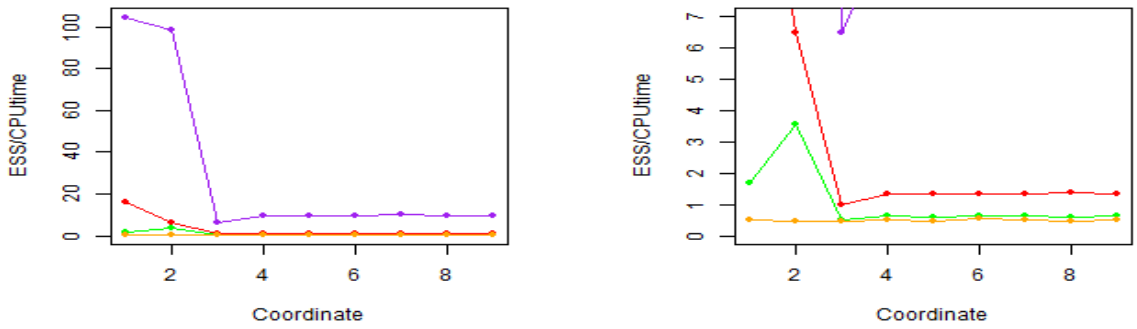


(b) "Banana-shaped" distribution

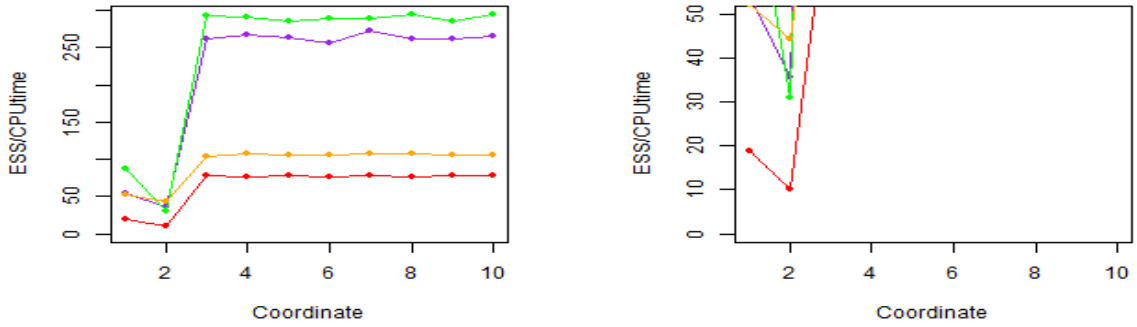


(c) Orange tree growth data

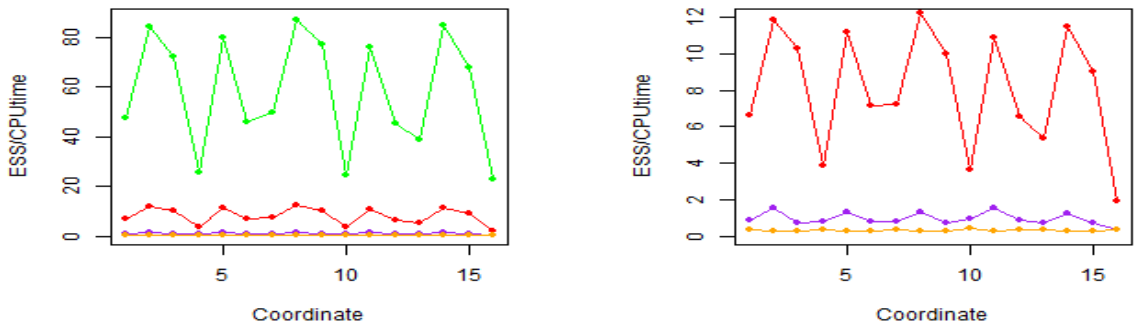
Figure 5.11: Comparison of ESS for different adaptive schemes. The red represents the ACMTM algorithm; the purple represents the AMwG algorithm; the green represents the SCAM algorithm; and the orange represents the AM algorithm. In each row, the right panel is a close up that identifies features that cannot be clearly seen in the left panel. ESS is calculated after averaging ACT over 50 independently replicated runs.



(a) Variance components model



(b) "Banana-shaped" distribution



(c) Orange tree growth data

Figure 5.12: Comparison of ESS/CPUtime for different adaptive schemes. The red represents the ACMTM algorithm; the purple represents the AMwG algorithm; the green represents the SCAM algorithm; and the orange represents the AM algorithm. In each row, the right panel is a close up that identifies features that cannot be clearly seen in the left panel. ESS is calculated after averaging ACT over 50 independently replicated runs.

Comparison between samplers is achieved by comparing the worst coordinate-wise ESS and ESS/CPUtime. To be precise, we compare the

$$\min_{1\leq k\leq d} ESS_k$$

and

$$\min_{1\leq k\leq d} \frac{ESS_k}{CPUtime}$$

for each algorithm. We see in Figure 5.11 that if we disregard the additional computational cost for the CMTM, adaptive CMTM is the best choice out of 4 adaptive algorithms in every examples we run. To account for the computational cost, we calculate the ESS/CPUtime, and the results are shown in Figure 5.12. Adaptive CMTM is not the best algorithm based on this measure for each individual example. But, if we look at all three examples together, it is hard to say which one is the best algorithm out of all. The SCAM algorithm is the best one for the Orange tree growth data, but it is close to the worst one for the Variance Components Model. The Adaptive Metropolis-within-Gibbs algorithm is the best algorithm for the Variance Components Model, but it clearly did a lot worse than the CMTM algorithm for the Orange tree growth data. The Adaptive Metropolis did worse than the CMTM algorithm in two our of three examples. Once again, while not uniformly dominating, the ACMTM seems to perform robustly in all examples thus minimizing the risk of using an inefficient sampler in a given example.

5.6 Conclusion and Discussion

It is known that adaptive algorithms can be highly influenced by initial values given to their simulation parameters and by the quality of the chain during initialization period, i.e. the period during which no modifications of the transition kernel take place. ACMTM is no

exception, but there certain of its features can be thought of as means towards a more robust behaviour. For instance, the fact that we can start with multiple proposals makes it less likely that all five will be poor choices for a given coordinate. The motivation for ACMTM was given by situations in which the sampler requires very different proposals across coordinates and across regions of the state space. In such situations, traditional adaptive samplers are known to fail unless special modifications are implemented (Craiu et al. 2009; Bai et al. 2011a), but even these tend to underperform when d is high.

The adaption mechanism is very rapid as, once an alarm is triggered, the scales can change in multiple of 2's. The adaption mechanism is also stable since modifications to the kernel occur only if over selection from one of the boundary scale proposals is detected. Thus, if proposal scales are not perfect but good enough, they wouldn't be changing under this adaptive design since it is hard to keep choosing one proposal distribution out of multiple proposal distributions if each of them is useful in a big enough region of the state space. In other words, once the adaption reaches a good level, the proposal distributions would be stable and would not be constantly changing.

A general recommendation has been made by Craiu et al. (2009) to run, at least for a while, a number of adaptive samplers in parallel in order to insulate against a poor exploration of the state space during the initialization period. This idea can be extended easily to the ACMTM, especially in this age in which parallel processing is the norm rather than the exception. The increase in CPU time is the price we pay for the added flexibility of having multiple proposals and the ability to dynamically choose the ones that fit the region of the space so that acceptance rate and mixing rates are improved. And while this tend to attenuate the ACMTM's efficiency dominance, one cannot find among the algorithms we used for comparison in this chapter one that is performing better *on average* even after taking CPU time into account. This makes ACMTM a safe choice for multivariate MCMC sampling.

Finally, it is the authors' belief that AMCMC samplers will be more used in practice if their motivation is intuitive and their implementation is easy enough. We believe that the ACMTM fulfills these basic criteria and further modifications can be easily implemented once new needs are identified.

Chapter 6

Conclusion

In this thesis, we examined adaptive MCMC methods and made some contributions by developing adaptive diagnostics and an adaptive algorithm and further generalizing conditions to ensure the convergence of a broad range of adaptive algorithms.

First, we introduced adaptive diagnostics to find out if the chain has been adapted enough so the further adaption does not bring much more efficiency gain for the algorithm. With these diagnostics, we developed a general-purpose MCMC algorithm which automatically tunes its proposal distribution. This algorithm takes on the finite adaption method, so we avoided proving the convergence of infinitely-adapting algorithm. Also, we expanded our algorithm so it can more effectively handle 'strongly multimodal' target distributions, which impose a risk for the chain to be stuck at one mode for a long period of time hurting efficiency. We have applied our algorithm to several MCMC examples. By comparing with the random walk Metropolis algorithm with proposal distributions not 'tuned' or in other words not suitable for the particular target distributions, we showed that our algorithm managed quicker convergence.

We also contributed to generalize the conditions developed to ensure convergence of an adaptive MCMC algorithm. Craiu et al. (2015) suggested several conditions that guarantees

the convergence of an adaptive MCMC algorithm. One of the conditions is the continuity of the target density function. We extended on this and proved that an adaptive MCMC algorithm with a 'combocontinuous' target density function still converges to its target distribution if it satisfies other conditions imposed by Craiu et al. (2015). The concept 'combocontinuity' is defined in Chapter 4.

Lastly, we developed an adaptive scheme for the component-wise multiple-try Metropolis (CMTM) algorithm. We first showed that CMTM selects a proposal distribution out of multiple proposal distributions based on the local shape of the target distribution around the current state. This characteristic gives the CMTM an edge when encountering an irregularly-shaped target distribution. However, if multiple proposal distributions given for the CMTM are all bad ones for a particular target distribution, having a choice does not improve the efficiency much. Thus, our adaptive component-wise multiple-try Metropolis (ACMTM) algorithm adapts the set of proposal distributions given, based on the observation that the worse probability distributions have lower chance of getting selected in the CMTM setting. We then of course proved the convergence of ACMTM algorithm. We compared the ACMTM with other adaptive algorithms (introduced in Chapter 2) on several MCMC examples and showed that our algorithm displayed more robust performance than the adaptive algorithms compared.

This thesis collects the contributions we have made to the advance of the MCMC methods. There are still a lot more adaptive algorithms to be examined and developed, which can possibly improve efficiency more than what we have done. Also, the conditions developed to ensure the convergence of an adaptive MCMC algorithm can possibly be more generalized and simplified. Beside adaptive methods, other MCMC problems still wait for improved solutions (e.g. convergence diagnostics, multimodal targets). We hope future researches further advances the field of MCMC, so the MCMC method can be more generally and easily applied in practice, aiding statistical modelings of real-world problems.

Bibliography

Andrieu, C. and Atchadé, Y. F. (2007). On the efficiency of adaptive MCMC algorithms. *Electronic Communications in Probability*, 12(33):336–349.

Andrieu, C. and Moulines, E. (2006). On the ergodicity properties of some adaptive Markov Chain Monte Carlo algorithms. *The Annals of Applied Probability*, 16(3):1462–1505.

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

Atchadé, Y. F. and Rosenthal., J. S. (2005). On adaptive Markov Chain Monte Carlo algorithms. *Bernoulli*, 11(5):815–828.

Bai, Y., Craiu, R. V., and Di Narzo, A. (2011a). Divide and C onquer: A mixture-based approach to regional adaptation for MCMC. *J. Comput. Graph. Statist.*, 20(1):63–79.

Bai, Y., Roberts, G. O., and Rosenthal, J. S. (2011b). On the containment condition for adaptive Markov chain Monte Carlo algorithms. *Advances and Applications in Statistics*, 21(1):1–54.

Box, G. E. P. and Tiao, G. C. (1973). *Bayesian inference in statistical analysis*. Addison-Wesely, Reading, MA.

Brooks, S., Gelman, A., Jones, G. L., and Meng, X., editors (2011). *Handbook of Markov Chain Monte Carlo*. Taylor & Francis.

Brooks, S. P. and Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics*, 7(4):434–455.

Casarin, R., Craiu, R. V., and Leisen, F. (2013). Interacting multiple try algorithms with different proposal distributions. *Statistics and Computing*, 23(2):185–200.

Craiu, R. V., Gray, L., Latuszynski, K., Madras, N., Roberts, G. O., and Rosenthal, J. S. (2015). Stability of adversarial markov chains, with an application to adaptive mcmc algorithms. *Annals of Applied Probability*, 25(6):3592–3623.

Craiu, R. V. and Lemieux, C. (2007). Acceleration of the multiple-try metropolis algorithm using antithetic and stratified sampling. *Statistics and Computing*, 17(2):109–120.

Craiu, R. V., Rosenthal, J. S., and Yang, C. (2009). Learn from thy neighbor: Parallel-chain adaptive and regional MCMC. *J. Amer. Statist. Assoc.*, 104(488):1454–1466.

Davies, O. L. (1967). *Statistical methods in research and production*. Oliver & Boyd, Edinburgh and London.

Draper, N. and Smith, H. (1981). *Applied regression analysis*. John Wiley & Sons, New York.

Fort, G., Moulines, E., and Priouret, P. (2011). Convergence of adaptive and interacting Markov chain Monte Carlo algorithms. *The Annals of Statistics*, 39(6):3262–3289.

Gaver, D. P. and O'Muircheartaigh, I. G. (1987). Robust empirical Bayes analyses of event rates. *Technometrics*, 29(1):1–15.

Gelfand, A. E. and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409.

Gelman, A., Roberts, G. O., and Gilks, W. R. (1996). Efficient Metropolis jumping rules. *Bayesian Statistics*, 5(42):599–607.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472.

George, E. I., Makov, U. E., and Smith, A. F. M. (1993). Conjugate likelihood distributions. *Scandinavian Journal of Statistics*, 20(2):147–156.

Geyer, C. J. and Johnson, L. T. (2014). *MCMC: Markov Chain Monte Carlo*. R package version 0.9-3. `http://CRAN.R-project.org/package=mcmc`.

Giordani, P. and Kohn, R. (2010). Adaptive independent Metropolis–Hastings by fast estimation of mixtures of normals. *Journal of Computational and Graphical Statistics*, 19(2):243–259.

Haario, H., Laine, M., Mira, A., and Saksman, E. (2006). DRAM: efficient adaptive MCMC. *Statistics and Computing*, 16(4):339–354.

Haario, H., Saksman, E., and Tamminen, J. (1999). Adaptive proposal distribution for random walk metropolis algorithm. *Computational Statistics*, 14(3):375–396.

Haario, H., Saksman, E., and Tamminen, J. (2001). An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242.

Haario, H., Saksman, E., and Tamminen, J. (2005). Componentwise adaptation for high dimensional MCMC. *Computational Statistics*, 20(2):265–273.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Lindstrom, M. J. and Bates, D. M. (1990). Nonlinear mixed effects models for repeated measures data. *Biometrics*, 46:673–687.

Liu, J. S., Liang, F., and Wong, W. H. (2000). The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134.

Lunn, D., Spiegelhalter, D., Thomas, A., and Best, N. (2009). The BUGS project: Evolution, critique, and future directions. *Statistics in Medicine*, 28(25):3049–3067. `http://www.openbugs.net`.

Madras, N. (2002). *Lectures on Monte Carlo methods*. volume 16 of Fields Institute Monographs. American Mathematical Society, Providence, RI.

Madras, N. e. (2000). *Monte Carlo Methods*. volume 26 of Fields Institute Communications. American Mathematical Society, Providence, RI.

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(6):1087–1092.

Meyn, S. P. and Tweedie, R. L. (1993). *Markov chains and stochastic stability*. Springer-Verlag, London.

Neal, P. J., Roberts, G. O., and Yuen, W. K. (2012). Optimal scaling of random walk Metropolis algorithms with discontinuous target densities. *The Annals of Applied Probability*, 22(5):1880–1927.

Neal, R. M. (2011). MCMC using ensembles of states for problems with fast and slow variables such as Gaussian process regression. *arXiv:1101.0387*.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. `http://www.R-project.org`.

Roberts, G. O., Gelman, A., and Gilks, W. R. (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120.

Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16(4):351–367.

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

Roberts, G. O. and Rosenthal, J. S. (2006). Harris recurrence of Metropolis-within-Gibbs and trans-dimensional Markov chains. *The Annals of Applied Probability*, 16(4):2123–2139.

Roberts, G. O. and Rosenthal, J. S. (2007). Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of Applied Probability*, 44(2):458–475.

Roberts, G. O. and Rosenthal, J. S. (2008). Variance bounding Markov chains. *The Annals of Applied Probability*, 18(3):1201–1214.

Roberts, G. O. and Rosenthal, J. S. (2009). Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367.

Rosenthal, J. S. (2004). Adaptive MCMC Java applet. `http://probability.ca/jeff/java/adapt.html`.

Rosenthal, J. S. (2007a). AMCMC: An R interface for adaptive MCMC. *Computational Statistics & Data Analysis*, 51(12):5467–5470.

Rosenthal, J. S. (2007b). The AMCMC package. http://probability.ca/amcmc.

Rosenthal, J. S. (2011). Optimal proposal distributions and adaptive MCMC. In Brooks, S., Gelman, A., Jones, G. L., and Meng, X., editors, *Handbook of Markov Chain Monte Carlo*, pages 93–112. Taylor & Francis.

Rosenthal, J. S. and Yang, J. (2016). Ergodicity of discontinuous adaptive MCMC algorithms. Submitted for publication. Available at http://probability.ca/jeff/ftpdir/dini.pdf.

Rudin, W. (1976). *Principles of mathematical analysis*. McGraw-Hill New York, 3rd edition.

Scheidegger, A. (2012). *adaptMCMC: Implementation of a generic adaptive Monte Carlo Markov Chain sampler*. R package version 1.1. http://CRAN.R-project.org/package=adaptMCMC.

Soetaert, K. and Petzoldt, T. (2010). Inverse modelling, sensitivity and Monte Carlo analysis in R using package FME. *Journal of Statistical Software*, 33(3):1–28.

Soetaert, K. and Petzoldt, T. (2014). *FME: A Flexible Modelling Environment for Inverse Modelling, Sensitivity, Identifiability, Monte Carlo Analysis*. R package version 1.3.1. http://CRAN.R-project.org/package=FME.

Tierney, L. (1994). Markov chains for exploring posterior distributions. *the Annals of Statistics*, pages 1701–1728.

Turro, E., Bochkina, N., Hein, A. M. K., and Richardson, S. (2007). BGX: a Bioconductor package for the Bayesian integrated analysis of Affymetrix GeneChips. *BMC bioinformatics*, 8(1):439–448.

Vihola, M. (2010a). Grapham: graphical models with adaptive random walk Metropolis algorithms. *Computational Statistics and Data Analysis*, 54(1):49–54.

Vihola, M. (2010b). The Grapham package. http://www.stats.ox.ac.uk/~mvihola/grapham/.

Vihola, M. (2012). Robust adaptive Metropolis algorithm with coerced acceptance rate. *Statistics and Computing*, 22(5):997–1008.

Yang, J. (2014). *atmcmc: Automatically Tuned Markov Chain Monte Carlo*. R package version 1.0. http://CRAN.R-project.org/package=atmcmc.

Yang, J., Craiu, R. V., and Rosenthal, J. S. (2016). Adaptive component-wise multiple-try metropolis sampling. Submitted for publication. Available at http://arxiv.org/pdf/1603.03510.pdf.

Yang, J. and Rosenthal, J. S. (2016). Automatically tuned general-purpose MCMC via new adaptive diagnostics. Submitted for publication. Available at http://probability.ca/jeff/ftpdir/jinyoung1.pdf.