An Approximate MCMC Method for Convex Hulls

by

Pengfei Wang

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Statistics

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

Abstract

Markov chain Monte Carlo (MCMC) is an extremely popular class of algorithms for computing summaries of posterior distributions. One problem for MCMC in the so-called Big Data regime is the growing computational cost of most MCMC algorithms. Most popular and basic MCMC algorithms, like Metropolis-Hastings algorithm (MH) and Gibbs algorithm, have to take the full data set into account in every iteration. In Big Data case, it is a fact that datasets of more than 100 GB are now fairly common. The running time of standard MCMC on such large datasets is prohibitively long.

To solve this problem, some papers develop algorithms that use only a subset of the data at each step to obtain an approximate or exact posterior distribution. Korattikara et al (2013) [11] merely *estimates* the transition probabilities of a typical MH chain using a subset of the data at each step of the chain, with some controllable error. The Firefly Monte Carlo (FLYMC) algorithm, presented by Maclaurin and Adams [12], augments the original dataset and only explicitly evaluates an "active" subset in each step. They show that the marginal distribution of the FLYMC algorithm at stationarity is in fact still equal to the posterior distribution of interest. However, Both of the above two papers and other literature in this thesis are restrained to a special kind of posteriors with "product-form" likelihoods. Such posteriors require all data points are conditionally independent and under the same likelihood.

However, what problem we want to solve is targeting a uniform distribution on a convex hull. In this case, "product-form" is not applicable. The reason why we focus on this problem is in statistics we sometimes face the problem to compute the volume of distributions which have a convex hull shape or their shape is able to transformed into a convex hull. It is impossible to compute via decomposing and reducing convex hulls of high dimension. According to Bárány et al in 1987 [2], the ratio of the estimated upper and lower bound of the volume of a certain convex hull is quite big. It is not possible to estimate the volume well, either. Fast-mixing Markov chains are basically the only way to actually do volume computations.

The initial work in this thesis is to define a data-augmentation algorithm along the lines of FLYMC. We also introduce an auxiliary random variable to mark subsets. However, as our situation is more complicated, we also have one more variable to help selecting subsets than FLYMC algorithm. For the extra variable, we utilize pseudo-marginal algorithm (PMMH), which allows us to replace interest parameter's distribution conditional on augmented variable by an estimator. Although our algorithm is not a standard case because our estimator is biased, bounds of the individual approximating measure of the parameter of interest is able to be directly translated into bounds of the error in the stationary measure of the algorithm.

After finishing an implementable algorithm, we then use two tricks including Locality Sensitive Hash function (LSH) and Taylor's expansion to improve the original algorithm. LSH helps raise the efficiency of proposing new samples of the augmented variable. Taylor's expansion is able to produce a more accurate estimator of the parameter of interest.

Our main theoretical result is a bound on the pointwise bias of our estimator, which results in a bound on the total error of the chain's stationary measure. We prove the total error will converge under a certain condition. Our simulation results illustrate this, and we use a large collection of simulations to illustrate some tips on how to choose parameters and length of chains in real cases.

Acknowledgements

I would like to thank University of Ottawa and my supervisor Dr. Aaron Smith who made this possible.

Dedication

This is dedicated to my parents who soupport me with their best as well as my best friends who always are in my side and cured my homesick, including Sun Xuying, Ouyang Feng, Wei Lingchen, Liu Binchao, Guo Xiaojun, Xie Shiqin, Liu Dong.

Table of Contents

List of Tables viii

List of Figures ix

1 Introduction 1

Nomenclature 1

2 Background 5

2.1 Monte Carlo method and Markov Chain 5

2.1.1 Monte Carlo method 5

2.1.2 Markov Chain 6

2.2 Metropolis-Hastings algorithm and Rejection sampling 8

2.2.1 Rejection Sampling 8

2.2.2 Metropolis-Hastings algorithm 10

2.3 Gibbs algorithm and Metropolis within Gibbs algorithm 11

2.3.1 Gibbs algorithm 11

2.3.2 Metropolis within Gibbs algorithm 13

2.4 Augmenting MCMC 13

2.4.1 Data augmentation scheme 14

2.4.2 Pseudo-Marginal Algorithm 14

2.5 Big Data Problem 16

2.5.1 Problems for Big Data 16

2.5.2 Austerity MCMC 17

2.5.3 Can PMMH Help? 17

2.6 Taylor's Expansion 17

2.7 Convex hull case 18

3 Algorithm **19**

3.1 Basic Notation 19

3.2 An Initial Algorithm 20

3.3 Practicable density on θ 20

4 Advanced Algorithm **24**

4.1 Alternative Proposal Distributions 24

4.1.1 Build a LSH 24

4.1.2 Adjust $W_k(\theta)$ 27

4.2 Taylor's expansion of $W(\theta)$ 28

5 Analysis of Algorithm 10 **32**

5.1 Approaching variable $W(\theta)$ 32

6 Simulation **36**

6.1 Default Setting 36

6.2 Evaluation of $W_k(\theta)$ 36

6.3 Evaluation of algorithms 39

6.4 Suggestions of choosing (k, n) 48

7 Conclusion **53**

APPENDICES **54**

References **54**

List of Tables

6.1 scale relationship 49
6.2 Various Coefficients 51

List of Figures

2.1 Rejection Sampling Diagram 9

4.1 How to assign weights 25

6.1 Evaluate $W_k(\theta)$ for typical θ 37
6.2 Cost of $W_k(\theta)$ 38
6.3 density 40
6.4 MSE when iterations rises 41
6.5 MSE when iterations rises 42
6.6 MSE when iterations rises 43
6.7 MSE with different k 44
6.8 MSE with different k 45
6.9 Tendency 46
6.10 MSE with different k 47
6.11 maximum and minimum of $W_k(\theta)$ with different k 48
6.12 Model Error with respect to relationship 50
6.13 Model Error with respect to coefficients 52

Chapter 1

Introduction

Bayesian Inference (BI) is a popular statistical method which is distinguished by its treatment of parameters as random variables. To do BI, one needs two ingredients: the usual family of probability distributions indexed by the parameter space (usually called the *model* or *likelihood function*), and also a probability distribution on the parameter space (usually called the *prior*). With this setup, one then studies the conditional distribution of the parameter given the data, which is called the *posterior* distribution.

We denote $p_\theta(\theta)$ a prior on parameter space Θ, $p_{x|\theta}(x|\theta) = L(x|\theta)$ a model, and $p_{\theta|x}(\theta|x)$ the associated posterior. Finally, we denote the joint distribution of (θ, x) by $p_{\theta,x}(\theta, x)$ and the two marginal distributions by $p_\theta(\theta)$ and $p_x(x)$ respectively. By Bayes's rule, we have

$$\begin{aligned} p_{\theta|x}(\theta|x) &= \frac{p_{\theta,x}(\theta, x)}{p_x(x)} \\ &= \frac{p_{x|\theta}(x|\theta)p_\theta(\theta)}{\int p_{x|\theta}(x|\theta)p_\theta(\theta)d\theta} \\ &\propto L(x|\theta)p_\theta(\theta) \end{aligned}$$

In practice, the integral $\int p_{x|\theta}(x|\theta)p_\theta(\theta)d\theta$ that appears in the above equation (sometimes called normalizing constant) may be prohibitively difficult to compute. When the normalizing constant can't be computed, a Bayesian statistician typically can't use calculus to compute most quantities of interest, such as expectations and probabilities, with respect to the posterior distribution.

Fortunately, there is a generic technique for estimating probabilities and integrals with respect to posterior distributions *even when the normalizing constant is unknown:* the Monte Carlo method. The idea here is to do simulations instead of applying Bayes's rule directly. If we are able to repeatedly generate independent samples from a probability distribution, the mean value of samples will approach the expectation of the probability distribution by the Central Limit Theorem (CLT) [15]. This process of estimating an integral by repeated sampling is called the Monte Carlo method.

In general, it is hard to efficiently generate Independent and identically distributed (i.i.d.) samples from a posterior distribution. A frequently-used alternative is to instead simulate a *Markov chain* $\{\theta_i\}_{i=1}^n$ with transition probability $K_{\theta^*|\theta_t}(\theta^*|\theta_t)$ whose equilibrium distribution is equal to the target distribution $\pi_\theta(\theta)$. Those samples are correlated rather than independent since they rely on their previous states. Even though they are not independent, under a certain condition called "ergodicity" the Strong Law of Large Numbers (SLLN) applies [22].

There are quite a lot of methods to implement MCMC. Metropolis-Hastings (MH) [6] is the most basic and frequently used variant. The MH algorithm requires two ingredients: a target distribution (in the case of BI, this is the posterior distribution) and a proposal distribution P. To run the MH algorithm, we need a new sample θ^* from a proposal distribution $P(\theta^*|\theta_t)$ in every iteration. The algorithm then updates θ_{t+1} with θ^* within acceptance probability $\alpha(\theta_t, \theta^*) = min(1, \frac{\pi_\theta(\theta^*)P(\theta_t|\theta^*)}{\pi_\theta(\theta_t)P(\theta^*|\theta_t)})$. Otherwise, the old sample θ_t is kept, that is, $\theta_{t+1} = \theta_t$.

When performing BI, our target distribution in MCMC is the posterior $p_{\theta|x}(\theta|\{x_i\}_{i=1}^n)$ conditional on the dataset $X = \{x_i\}_{i=1}^n$. The corresponding transition probability $K_{\theta^*|\theta_t}(\theta^*|\theta_t)$ also depends on the data and can be written $K_{\theta^*|\theta_t}(\theta^*|\theta_t, \{x_i\}_{i=1}^n)$. For typical MCMC algorithms, including MH, every iteration has to take the full data set into account. In MH, the full dataset is used every time that we evaluate the acceptance probability, which requires us to compute the quotient $\frac{p_{\theta|X}(\theta^*|X)}{p_{\theta|X}(\theta_t|X)}$. This quotient can also be written as $\frac{L(X|\theta^*)p_\theta(\theta^*)}{L(X|\theta_t)p_\theta(\theta_t)}$. Suppose our observation $\{x_i\}_{i=1}^n$ given θ are conditionally independent. Then the likelihood is $L(\{x_i\}_{i=1}^n|\theta) = \prod_{i=1}^n L(x_i|\theta)$ Posteriors with such likelihoods can be written as a product-form $p(\theta|X) = p(\theta)\prod_{j=1}^n p(X_j|\theta)$. In the Big Data case, n is extremely large. It is a fact that datasets of more than 100 GB are now fairly common. The running time of standard MCMC on such a large dataset is prohibitively long. The computational cost of computing the posterior distribution for i.i.d data scales at least linearly with respect to the data size. For many non-i.i.d posterior models such as Gaussian Process regression, the scaling is much higher perhaps quadratic or cubic, e.g. we need invert an $n \times n$ matrix, which naively takes time cubic in n. For some special cases, it is possible to get down to quadratic if processing carefully. But the scale is hard to continually reduce.

To speed up processing data with product-form posteriors, some papers consider to use only partial data at each step. Korattikara et al in 2013 put forward an algorithm for Big Data sets. If drawing a large number of samples is cheap and reducing the variance is fast, they are willing to tolerate a controllable small bias produced by only using partial data at each step to speed up their loop. They use a hypothesis test by comparing the difference of log-likelihoods of partial data to decide if accept a new sample. This action replaces acceptance probability of the MH algorithm. Even though there is a bias, it can be compensated for by increasing the cardinality of subsets until a prescribed confidence level is reached. In extreme case, they do the exact sampling.

Another class of algorithms is "data augmentation" put forward by Martin et al in 1987 [19]. It augment the original state space with auxiliary random variables defined as we like.

Then sampling the parameter of interest will condition on the auxiliary random variable as well as the whole dataset. Perhaps surprisingly, it is sometimes possible to sample from the *exact* posterior even when using only a subsample of the data at every step. The Firefly Monte Carlo (FLYMC) algorithm, presented by Maclaurin and Adams [12], modifies the original dataset to an augmented data by introducing an auxiliary indicator variable $Z \subset \{0,1\}^n$ that represents the "active" subset of the dataset. They then construct a Markov chain (Θ_t, Z_t) with a new joint distribution $\pi_{\theta,Z}(\theta, z)$ on this state space. They show that the marginal distribution $\pi_\theta(\theta|\{x_i\}_{i=1}^n)$ at stationarity is in fact still equal to the posterior distribution of interest.

This data augmentation algorithm uses the MH algorithm to update alternately between sampling θ and Z under their own conditional distribution $p_{\theta|z,x}$ and $p_{z|\theta,x}$. $p(\theta|z,x)$ depends *only* on the data within the current subset indexed in z. $p_{z|\theta,x}$ *only* depends on θ but related to both of the likelihood function $L(x|\theta)$ and a given lower "bound" function $B(\theta)$ which is much easier to compute than $L(x|\theta)$ but still close to $L(x|\theta)$. FLYMC switches subsets in iterations so as to maintain the correct stationary distribution by leaving the true full-data posterior distribution invariant. Although the MH algorithm uses all observations in each step, only a small part of data is used to evaluated explicitly. The rest are approximated by the bound function $B(\theta)$.

Andrieu and Roberts in 2009 [1] also developed a new algorithm named Pseudo-Marginal algorithm (PMMH) based on the data augmentation. They thought this algorithm enabled to produce correct stationary distribution by replacing a complicated conditional distribution $p_{\theta|Z}(\theta|z)$ (abbreviated as $p(\theta)$) with its *unbiased estimator* $\hat{p}(\vartheta)$ in MH algorithm. Variable ϑ and proposal distribution $\hat{p}(\vartheta)$ are used to replace θ and $p(\theta)$ respectively. PMMH algorithm is not only proved by Tran et al (2016) [21] to save more time than data augmentation scheme, but also makes acceptance probabilities practically computable. It is quite easy to write down an unbiased estimator of a log-likelihood based on a small sample of a full dataset. However, the work of Jacob et al in 2015 [9] showed that unbiased estimator of likelihoods is not possible.

All above methods are restrained to product-form posteriors. Can we remove the requirement of product-form but at the same time slow down the grow of computational cost with respect to the data size like the literature mentioned before. The concrete case we want to solve is targeting a uniform distribution on a convex hull. The reason is in statistics, we sometimes face the problem to compute the volume of distributions which have a convex hull shape or whose shape is able to transformed into a convex hull. According to much literature in computer science, computing the volume of a convex hull is difficult. In 1987, Bárány et al [2] showed estimated upper and lower bound of the volume is too big to be used. Simonovits et al [17] pointed out deterministic methods is not practical for the high dimension case. One more efficient solution is to use fast-mixing Markov chains for sampling a uniform on the hull. This is basically the only way to achieve the goal. Hence, in order to solve volume computation problem, we try to find a good MCMC method of targeting a uniform distribution on a convex hull in Big Data case.

In Chapter 2 we present a more detailed background of theories and algorithms we used. It includes the idea of how to define a Monte Carlo Markov Chain and some tricks of MCMC method's variants. We also present more details about convex hull case of interest. We explain what difficulties we would face.

In Chapter 3 we construct a variant algorithm of MCMC with details to achieve our goal. The rough version is an analog of FLYMC. We augment our dataset $\{X_i\}_i^N$ by bringing in a variable S to determine subsets of original dataset for iteration. As no surprise, the rough version has a complicated joint distribution $\pi(\theta, S)$ with corresponding conditional probability on θ, which is incredibly time-consuming to generate. To speed up the algorithm, instead of accurately estimating the acceptance probability, we approach it through a biased but non-negative estimator $W(\theta)$ under the PMMH framework.

Although we already obtain an improved algorithm in Chapter 3, there is some space to improve it. In Chapter 4 we increase efficiency and accuracy of the algorithm by equipped with two tools, Locality Sensitive Hash function (LSH) and Taylor's expansion. LSH helps raise the efficiency of proposing samples of S. Since pseudo-marginal variable is given, Taylor's expansion is applied on it to get a more accurate estimator of the parameter of interest. A more accurate pseudo-marginal variable will definitely reduce errors of the algorithm. Besides those main algorithms to construct MCMC, many sub-algorithms to implement the tools are also included in this chapter.

In Chapter 5, we theoretically discuss the bound of the difference of pseudo-marginal variable and its estimator. We successfully prove the total error on the whole convex hull converges with its parameter increasing.

Chapter 6 gives many simulations to support our algorithms and display the performance of algorithms and the pseudo-marginal variable. In terms of the pseudo-marginal variable, we evaluate its convergence and cost. As for our main algorithms, we plot their densities and study how to reduce the mean-squares errors of difference of the true density and the estimated one. In the end of this chapter, we give some tips how to choose parameters and length of chains in real cases.

Chapter 2

Background

This chapter provides an overview of necessary background mainly including how to implement Monte Carlo Markov Chain. We provide definitions and properties of Markov chain. We also introduce basic and advanced algorithms. In the end, we present what a situation our algorithm aims at.

2.1 Monte Carlo method and Markov Chain

The name of MCMC suggests it is composed of two components i.e. Monte Carlo method and Markov Chain. This section introduces basic principles of the two methods.

2.1.1 Monte Carlo method

The Monte Carlo method was invented by Los Alamos Scientific Laboratory and formally presented by Metropolis et al (1949) [13]. After more than half a century, it is developed into a broad class of computational algorithms relying on repeated random sampling to obtain numerical solutions. It is frequently utilized in the case where integrals are hard to compute directly. The Monte Carlo method estimates an integral $\int h(x)dP(x)$ by

$$\frac{1}{n}\sum_{i=1}^{n} h(X_i), \tag{2.1.1}$$

where $X_1, \ldots, X_n \overset{i.i.d.}{\sim} P$.

The Law of Large Number (LLN) [22] implies that the sum (2.1.1) converges to $\int h(x)dP(x)$ as more trials are performed. Because of the LLN, we can safely estimate the expected value of a random variable with the sample average. By the result of repeating sampling from a given and easily computed probability distribution function, the integral can be approached with a low cost because we avoid complicated or even impossible calculation.

2.1.2 Markov Chain

Markov chain is a stochastic model named after Andrey Markov [4]. A Markov Chain is a sequence of random variables $X_1, X_2, \ldots$ that satisfies the following "memoryless" property

$$P(X_{t+1} \in A | X_1 = x_1, X_2 = x_2, ..., X_t = x_t) = P(X_{t+1} \in A | X_t = x_t) \tag{2.1.2}$$

for all $t \in \mathbb{N}$, measurable A and sequence $x_1, \ldots, x_t$.

We call a Markov chain time-homogeneous if for an arbitrary time point n

$$P(X_{n+1} \in A | X_n = B) = P(X_2 \in A | X_1 = B) \tag{2.1.3}$$

That says the transition of Markov chain is independent with time n. In this thesis, we only discuss about time-homogeneous Markov chain.

The possible values of X compose a set $\mathbb{S}$ called the state space of the chain. Every different value is an individual state. The state space can be countable or not countable. In our thesis the state space is a product of a continuous state space and a finite state space. The Markov chain can enter into an arbitrary state or stay in the same state with a certain probability after a period of time. But only one state change may occur after a unit time interval. The transition probability is a conditional probability like Equation 2.1.2. The transition probability is also called a one-step transition probability. Equation 2.1.2 reveals the next state is only determined by the present state.

There is a tool to describe all probabilities of transition between all states after a unit time interval. If state space is finite and countable, we can build a square matrix $P = \{a_{i,j}\}_{n \times n}$ called transition probability matrix to describe the transition between two states i and j. Element $a_{i,j}$ records probability $P_{j|i}(j|i) \equiv P(X_2 = j | X_1 = i)$ of going from state i to state j ($i = j$ is acceptable). In our thesis, the probability distribution function P of generating all states is given. We can construct a series of conditional probability distribution function also denoted as $P_{j|i}(j|i)$ of going from state i to state j ($i = j$ is also acceptable).

Once we know the transition probability matrix or function $P_{j|i}(j|i)$, we can uniquely build a Markov chain. Denote the n-step transition probability given transition probability matrix or function $P_{j|i}(j|i)$ as $P^{(n)}_{j|i}(j|i)$ satisfying following condition: for any k such that $0 < k < n$,

$$P^{(n)}_{j|i}(j|i) = \sum_{r \in \mathbb{S}} P^{(k)}_{r|i}(r|i) P^{(n-k)}_{j|r}(j|r)$$

The condition is also known as the Chapman-Kolmogorov equation [4]. At time t, using the shorthand $P_{ij} = P(X_{t+1} = j | X_t = i)$. Accordingly $P^{(n)}_{ij} = P^{(n)}_{j|i}(j|i)$. We then present some necessary properties of a Markov chain.

- A state i has period $k \geq 1$ if any chain starting at and returning to state i with positive probability must take a number of steps divisible by k. If $k = 1$, then the state is known as aperiodic, and if $k > 1$, the state is known as periodic. If all states are aperiodic, then the Markov chain is known as aperiodic.

- A Markov chain is known as irreducible if there exists a chain of steps between any two states that has positive probability.
- An absorbing state i is a state for which $P_{ii} = 1$.
- A state is known as recurrent or transient depending upon whether or not the Markov chain will eventually return to it. A recurrent state is known as positive recurrent if it is expected to return within a finite number of steps and null recurrent otherwise.
- A state is known as ergodic if it is positive recurrent and aperiodic. A Markov chain is ergodic if all its states are.

A Markov chain is said to be reversible if there is a probability distribution π over its states such that $\pi_i P_{ij} = \pi_j P_{ji}$ from all states i and j. This condition is known as the detailed balance condition. Such a probability distribution π is also called stationary distribution or equilibrium distribution of the chain. If $j \in \mathbb{S}$, π has following properties:

1. $0 \leq \pi_j \leq 1$
2. $\sum_{j \in \mathbb{S}} \pi_j = 1$
3. $\pi_j = \sum_{i \in \mathbb{S}} \pi_i p_{ij}$

Stationary distribution π is the probability distribution function $P(\cdot)$ of all states. If and only if all of its states are positive recurrent, an irreducible chain has a positive stationary distribution which means for $\forall i$, $\pi_i > 0$. In this case, π is unique. Further, if the positive recurrent chain is both irreducible and aperiodic, it is said to have a limiting distribution. For $\forall i$ and j, $lim_{n \to \infty} p_{ij}^{(n)}$ exist and is only relative to state j. With above properties, assume the initial distribution π_0 is given. After n-step transition, the current distribution becomes $\pi_0 P^n$. We have $lim_{n \to \infty} \pi_0 P^n = \pi$. As long as we know the transition probability matrix or function P, it is easily implementable to obtain samples approximately from stationary distribution after limited transition steps. See Geyer [8] for a review of more basic MCMC theory.

When we only have one random variable, Algorithm 2.1.2 compose a set $\{x_n, x_{n+1}, ...x_{n+N}\}$ which approximately comes from stationary distribution π.

Algorithm 1 MCMC with one parameter of interest

Initialize wanted sample size N and burning-in sample size n.
Input transition probability function (or matrix) P and a starting value x_0.
for $t = 0$ to $n + N - 1$ **do**
  Sample $x_{t+1} \sim P(\cdot|x_t)$.
end for
Output sample set $\{x_n, x_{n+1}, ...x_{n+N}\}$.

Based on this sample set, Monte Carlo method can be applied to estimate the integral $\mathbb{E}(h)$ by $\frac{1}{N+1}\sum_{j=n}^{n+N} h(x_j)$. In practice, burning-in sample size n are not decided solely in advance. Instead, we are only provided with iteration times. After we obtain chains, we then diagnose how many samples to discard or burn-in.

2.2 Metropolis-Hastings algorithm and Rejection sampling

According to Algorithm 2.1.2 as long as we are able to repeat sampling under transition probability function (or matrix) P, we can easily generate a sample set approximately from the stationary distribution π. However, sometimes P is computationally or analytically intractable so that Algorithm 2.1.2 is not implementable in practice. For example, $P(\cdot|x_t)$ has a quite complicate expression so that we are not able to find it inverse function. Fortunately, MCMC provides a framework to deal with different cases. Among them, Metropolis-Hastings (MH) algorithm and Gibbs algorithm are the most popular and basic. In this section, we introduce MH algorithm.

2.2.1 Rejection Sampling

Variants of MCMC are designed with different way to generate samples, but many of them are based on a technique called *Rejection Sampling*. More detailed introduction can be found in [5]. Assume our target distribution $p(x)$ is hard to generate samples. Let $q(x)$ be a proposal distribution which is familiar and easy to sample from, and for which there exists a constant $k < \infty$ satisfying $p(z_0) \leq kq(z_0)$ for all z_0 (see Figure 2.2.1, created by Liu[10]).

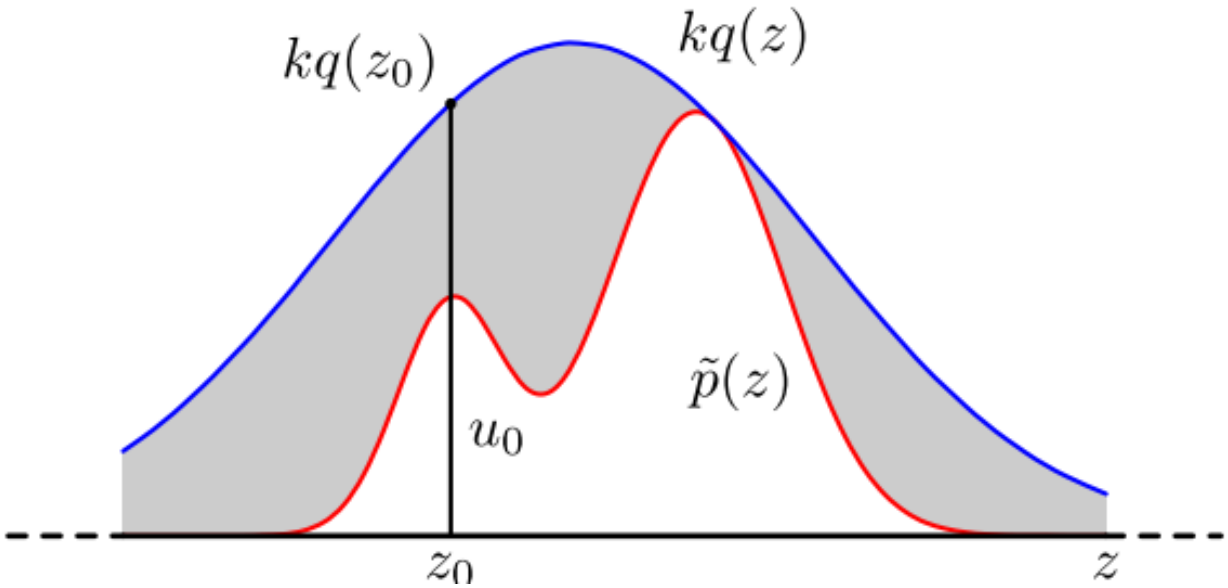

Figure 2.1: Rejection Sampling Diagram

Then do the Algorithm 2 until we obtain n samples:

Algorithm 2 Iteration of Rejection Sampling

while $i \leq n$ **do**
  Sample $x_i \sim q(x)$.
  Compute acceptance probability $\alpha = \frac{p(x_i)}{kq(x_i)}$.
  Sample $u \sim Uniform(0, 1)$.
  if $u \leq \alpha$ **then**
    Keep x_i as a sample and $i = i + 1$
  end if
end while

From Rejection Sampling Diagram 2.2.1, we can obviously find it is more likely to accept samples where blue and red lines are close. By repeated sampling, the density plot of samples will be nearly close to true density plot (red line). Since random variable u is under $uniform(0, k)$, $P(u > \alpha) = k * \frac{p(x)}{kq(x)} = \frac{p(x)}{q(x)}$. Therefore, the density of samples is $q(x) * P(u > \alpha) = p(x)$.

2.2.2 Metropolis-Hastings algorithm

MH algorithm is similar to Rejection Sampling by sampling from a proposal distribution. For a random variable $X \in R^m$ with a joint distribution π and a reversible proposal kernel $Q(X_i, X_j)$ which stands for the transition probability from state X_i to X_j, MH algorithm (Algorithm 3) defines a kernel

$$K_{MH}(X_i, X_j) = Q(X_i, X_j)\alpha(X_i, X_j)$$

where $\alpha(X_i, X_j) = min\{1, \frac{\pi(X_j)Q(X_j,X_i)}{\pi(X_i)Q(X_i,X_j)}\}$. $\alpha(X_i, X_j)$ is the acceptance probability defined similarly to Rejection Sampling. Note that the MH algorithm is determined by its proposal kernel and its target distribution; we will use this fact several times.

With above definition, we can present a theorem as following:

Theorem 2.2.1. *For two arbitrary points X_i and X_j, if proposal distribution is symmetry $Q(X_i, X_j) = Q(X_j, X_i)$, we have $\pi K_{MH} = \pi$.*

It is easy to prove Theorem 2.2.1 by proving detailed balance equation.

Proof.

$$\begin{aligned}
\pi(X_i)K_{MH}(X_i, X_j) &= \pi(X_i)Q(X_i, X_j)\alpha(X_i, X_j) \\
&= \pi(X_i)Q(X_i, X_j)min\{1, \frac{\pi(X_j)Q(X_j, X_i)}{\pi(X_i)Q(X_i, X_j)}\} \\
&= min\{\pi(X_i)Q(X_i, X_j), \pi(X_j)Q(X_j, X_i)\} \\
&= \pi(X_j)Q(X_j, X_i)min\{1, \frac{\pi(X_i)Q(X_i, X_j)}{\pi(X_j)Q(X_j, X_i)}\} \\
&= \pi(X_j)Q(X_j, X_i)\alpha(X_j, X_i) \\
&= \pi(X_j)K_{MH}(X_j, X_i)
\end{aligned}$$

□

MH algorithm is listed below:

Algorithm 3 Metropolis-Hastings algorithm

Input iteration time N and initialize x_0.
for $t = 0$ to $n + N - 1$ **do**
  Sample $x^* \sim Q(x_t, \cdot)$.
  Sample $u \sim Uniform(0, 1)$.
  Calculate $\alpha(x_t, x^*) = min\{1, \frac{\pi(x^*)Q(x^*, x_t)}{\pi(x_t)Q(x_t, x^*)}\}$
  if $u \leq \alpha(x_t, x^*)$ **then**
    Set $x_{t+1} = x^*$.
  else
    Set $x_{t+1} = x_t$.
  end if
end for
Output sample set $\{x_0, x_1, ...x_N\}$.

2.3 Gibbs algorithm and Metropolis within Gibbs algorithm

2.3.1 Gibbs algorithm

In some situations, it is not too hard to sample exactly from the one-dimensional *conditional* distributions of a high-dimensional target distribution π. In such situations, there is a special case of the MH algorithm called the *Gibbs sampler* that can be very efficient.

Assume we have a variable vector of interest $X = (X_1, X_2, ..., X_m)$ with a joint distribution π. For each component X_i, denote $X_{-i} = (X_1, ..., X_{i-1}, X_{i+1}, ..., X_m)$. Denote the conditional distribution $P(X_i|X_1, X_2, \cdot, X_{i-1}, X_{i+1}, \cdot, X_m)$ of X_i given everything else by π_i.

As a special case of MH algorithm, Gibbs algorithm also needs proposal distributions to sample. The difference is that in Gibbs algorithm we only update one component X_i rather than the whole vector X in one loop step. Each X_i has its own proposal distribution

$$Q_i(X_i, X_{-i}; X_i^*, X_{-i}) = \pi_i(X_i^*|X_{-i})$$

. The order of updating components can be fixed or random. Regardless of updating ways, all draws from a proposal distribution are accepted.

With these changes, Theory 2.2.1 still works. That is:

Theorem 2.3.1. *A single conditional distribution π_i can be targeted by a component transition kernel*

$$K_{G,i}(X_i, X_{-i}; X_i^*, X_{-i}) = Q_i(X_i, X_{-i}; X_i^*, X_{-i})$$

Then, we have $\pi_i K_{G,i} = \pi_i$.

To prove it, we only need to show the acceptance probability in this case is always 1.

Proof.

$$\begin{aligned}\alpha(X_i, X_{-i}; X_i^*, X_{-i}) &= min\{1, \frac{\pi_i(X_i^*|X_{-i})Q_i(X_i^*, X_{-i}; X_i, X_{-i})}{\pi_i(X_i|X_{-i})Q_i(X_i, X_{-i}; X_i^*, X_{-i})}\}\} \\ &= min\{1, \frac{\pi_i(X_i^*|X_{-i})\pi_i(X_i|X_{-i})}{\pi_i(X_i|X_{-i})\pi_i(X_i^*|X_{-i})}\}\} \\ &= min\{1, 1\} \\ &= 1\end{aligned}$$

When targeting π_i with Q_i, set kernel

$$\begin{aligned}K_{G,i}(X_i, X_{-i}; X_i^*, X_{-i}) &= Q_i(X_i, X_{-i}; X_i^*, X_{-i})\alpha(X_i, X_{-i}; X_i^*, X_{-i}) \\ &= Q_i(X_i, X_{-i}; X_i^*, X_{-i})\end{aligned}$$

By Theory 2.2.1 we have $\pi_i K_{G,i} = \pi_i$. □

We define Gibbs kernel $K_G = \sum_{i=1}^m w_i K_{G,i}$, where the weights $w_1, \ldots, w_n \geq 0$ satisfy $\sum_{i=1}^m w_i = 1$.

Then we have

Theorem 2.3.2. $\pi K_G = \pi$

Gibbs algorithms can be written with components updated in either a random order or a fixed order. Theorem 2.3.2 is about the randomly-updated Gibbs algorithm.

Proof. This is a quick calculation. Since the conditionals π_i are all preserved by each $K_{G,i}$, so is the full distribution π. We can then write:

$$\begin{aligned}\pi K &= \pi(\sum_{i=1}^m w_i K_{G,i}) \\ &= \sum_{i=1}^m w_i \pi K_{G,i} \\ &= \sum_{i=1}^m w_i \pi \\ &= \pi.\end{aligned}$$

□

For completeness, we list the Gibbs algorithm with a fixed updating order as below:

Algorithm 4 Gibbs algorithm

Input iteration time N and initialize x_0.
for $t = 0$ to N **do**
  Sample $x_1^{(t+1)} \sim \pi(X_1|x_{-1}^{(t)})$.
  Sample $x_2^{(t+1)} \sim \pi(X_2|x_{-1,-2}^{(t)}, x_1^{(t+1)})$.
  Sample $x_3^{(t+1)} \sim \pi(X_3|x_{-1,-2,-3}^{(t)}, x_1^{(t+1)}, x_2^{(t+1)})$.
  ...
  Sample $x_m^{(t+1)} \sim \pi(X_m|x_{-m}^{(t+1)})$.
  Set $x^{(t+1)} = (x_1^{(t+1)}, x_2^{(t+1)}, ..., x_m^{(t+1)})$
end for
Output sample set $\{x^{(0)}, x^{(1)}, ...x^{(N)}\}$.

See *e.g.* Schmeiser et al [16] for a proof that this algorithm also has the correct stationary measure.

2.3.2 Metropolis within Gibbs algorithm

Although conditional distributions are much easier to deal with compared with a joint distribution, sometimes they are still analytically or computationally intractable. It is worthwhile to try to combine advantages of MH algorithm and Gibbs algorithm since the usage of acceptance probability in MH algorithm can avoid directly drawing samples from a troublesome conditional distribution and instead draw from a convenient proposal distribution. The combination is called Metropolis-Hastings within Gibbs. Chib and Greenberg [6] and Tierney [20] proved the stationary distribution still holds. We have such a theory:

Theorem 2.3.3. *For $i \in \{1, 2, \ldots, m\}$, let $K_{G,i}$ be any transition kernel with stationary measure π_i. If X_i's updating probability satisfies Theorem 2.3.1, then $K_G = \sum_{i=1}^m w_i K_{G,i}$ has stationary measure π.*

Proof. The proof is identical to that of Theorem 2.3.2. □

In particular, we can use MH kernels for some or all of the kernels $K_{G,i}$.

2.4 Augmenting MCMC

MH sampler, Gibbs sampler and Metropolis within Gibbs algorithm provide a way to solve the integral problem but they do not promise the best efficiency. In some cases, it turns

out to be easier to solve the problem of interest by embedding it in a larger but simpler problem. This is the basic idea behind *data-augmentation* schemes. We present here both the generic data-augmentation scheme and a special case, the pseudo-marginal algorithm, that is most relevant for the thesis.

2.4.1 Data augmentation scheme

The data augmentation idea is not really a new algorithm, like the MH, Gibbs, and MH-within-Gibbs algorithms described above. It is instead a new perspective on the original problem of estimating integrals with respect to a measure π. Tanner and Wong [19] introduced this idea and showed how to implement it.

We begin with a target measure π of interest on a sample space Θ. We then consider an *augmented* measure $\hat{\pi}$ on an *augmented* sample space $\Theta \times Z$ that satisfies the following marginalization condition: for $(\theta, z) \sim \hat{\pi}$, the marginal distribution of θ is π. We then make the following simple observation: if $(\theta_1, z_1), (\theta_2, z_2), \ldots$ is a Markov chain with stationary measure $\hat{\pi}$, then we can estimate an integral $\pi(h)$ using just the marginal chain: $\pi(h) \approx \frac{1}{N} \sum_{i=1}^{n} h(\theta_i)$. In practice, the main difficulty with this algorithm is finding an augmented space Z that is well-adapted to the problem and makes sampling easier.

Mathematically, there is no distinction between sampling from π and sampling from $\hat{\pi}$ - any of the earlier approaches to sampling from π can be used to sample from $\hat{\pi}$, and the algorithms don't know that we are really only interested in the marginal distribution of the original variable. In many cases, the augmented state space is chosen so that it is easy to sample from the conditional distributions $p_{\Theta|X,z}(\theta|X, z)$ and $p_{Z|\Theta,X}(z|\theta, X)$. In this case, we can just use the two-state Gibbs algorithm, leading to Algorithm 5:

Algorithm 5 Data Augmentation algorithm

Input iteration time N and dataset X
Initialize θ_0 and z_0.
for $t = 0$ to $N - 1$ **do**
  Sample $z_{t+1} \sim p_{Z|\Theta,X}(\cdot|\theta_t, X)$.
  Sample $\theta_{t+1} \sim p_{\Theta|Z,X}(\cdot|z_{t+1}, X)$.
end for
Output sample set $\{\theta_0, \theta_1, ..., \theta_N\}$.

Theorem 2.3.1 guarantees that this algorithm has the correct stationary measure.

2.4.2 Pseudo-Marginal Algorithm

The Pseudo-Marginal Metropolis-Hastings Algorithm (PMMH) is an interesting special case of the data-augmentation idea, developed in Andrieu and Roberts (2009) [1] to deal

with certain types of intractable likelihoods. To set up the idea, we consider as in Section 2.2.2 a proposal distribution π and a proposal distribution Q, with densities p and q respectively. The MH algorithm in Section 2.2.2 required us to be able to compute ratios of the form $\frac{p(\theta_1)}{p(\theta_2)}$ (that is, we must be able to evaluate a function $f(\theta) = Cp(\theta)$ for some perhaps-unknown constant C that does not depend on $\theta \in \Theta$). Sometimes this is quite difficult. The insight of [1] is that one doesn't need the exact function f. Instead, it is enough to construct a family of distributions $\{\mu_\theta\}_{\theta\in\Theta}$ with the property that for $Z \sim \mu_\theta$,

$$E[Z] = Cp(\theta); \tag{2.4.1}$$

The usual MH algorithm corresponds to the special case that μ_θ is a point mass at $f(\theta)$.

It is not immediately obvious how best to *use* the distributions μ_θ, but [1] lead to Algorithm 6:

Algorithm 6 Pseudo-Marginal Metropolis-Hastings algorithm

Input iteration time N, dataset X.
Initialize θ_0 and z_0.
for $t = 0$ to $N - 1$ **do**
  Sample $\vartheta^* \sim q(\cdot|\theta_t)$.
  Sample $z^* \sim \mu_{\theta^*}$.
  Sample $u \sim Uniform(0, 1)$.
  Calculate $\alpha(\theta_t, z_t; \vartheta^*, z^*) = min\{1, \frac{z^* q(\theta_t|\vartheta^*)}{z_t q(\vartheta^*|\theta_t)}\}$
  if $u \leq \alpha(\theta_t, z_t; \vartheta^*, z^*)$ **then**
    Set $\theta_{t+1} = \vartheta^*$ and $z_{t+1} = z^*$.
  else
    Set $\theta_{t+1} = \theta_t$ and $z_{t+1} = z_t$.
  end if
end for
Output sample set $\{\theta_0, \theta_1, ...\theta_N\}$.

Note that this is a data-augmentation algorithm on the augmented state space $\Theta \times [0, \infty)$. The main result is that the output of this algorithm has the correct stationary measure, which we summarize as:

Theorem 2.4.1. *Denote by $\hat{\pi}$ the stationary measure of Algorithm 6. Let $(\theta, z) \sim \hat{\pi}$; then $\theta \sim \pi$.*

In certain special cases, it is much easier to find *some* family of measures $\{\mu_\theta\}_{\theta\in\Theta}$ satisfying Equation (2.4.1) than it is to evaluate the likelihood. In these situations, PMMH can be much faster than MH (and MH could be completely intractable). In this thesis, we use it for a very different reason. In our situation, it is too hard to find a family of measure μ_θ that *exactly* satisfies Equation (2.4.1). However, even for these "approximate" measures,

we can use Theorem 2.4.1 to write down the *exact* stationary measure of the associated PMMH algorithm. This allows us to translate bounds on the bias of the individual approximating measure μ_θ directly into bounds on the error in the stationary measure of the algorithm; without the PMMH algorithm, this would usually be very hard.

2.5 Big Data Problem

Prohibitive cost of computation is a critical fact into practical application of MCMC. This section discusses why computational cost is not affordable for the Big data case and relates this to the previous algorithms.

2.5.1 Problems for Big Data

In Bayesian Inference, we apply Algorithm 3 and 4 with stationary distribution π given by some posterior distribution $p_{\theta|x}(\theta|X)$ with respect to some dataset $X = \{x_i\}_{i=1}^n$. Accordingly, the transition probability P of current state in Algorithm 2.1.2 is conditional on both its previous state and the dataset. To reflect this in our notation, we can write $\theta_{t+1} \sim P(\cdot|\theta_t, X)$, and the acceptance probability $\alpha(\theta_t, \theta^*)$ in MH algorithm becomes

$$min\{1, \frac{p_{\theta|X}(\theta^*|X)q(\theta^*|\theta_t, X)}{p_{\theta|X}(\theta_t|X)q(\theta_t|\theta^*, X)}\}$$

where q is an easily available proposal probability with a similar definition of Q in Algorithm 3. In Gibbs algorithm, the conditional distribution of every component should include dataset X. For example, for a parameter vector $(\theta, z) \in R^2$, probabilities of updating components should be $p_{\theta|z}(\cdot|z, X)$ and $p_{z|\theta}(\cdot|\theta, X)$ respectively.

Typically, every step of these algorithms must take into account the full dataset (though there are exceptions). Take MH algorithm as an example. We have to evaluate acceptance probability and compute the quotient $\frac{p_{\theta|X}(\theta^*|X)}{p_{\theta|X}(\theta_t|X)}$ in every iteration. We can rewrite this as $\frac{L(X|\theta^*)\pi_\Theta(\theta^*)}{L(X|\theta_t)\pi_\Theta(\theta_t)}$ with likelihood $L(X|\theta)$ and prior $\pi_\Theta(\theta)$. Suppose our observation $\{x_i\}_{i=1}^n$ given θ are conditionally independent. Then the likelihood is $L(X|\theta) = \prod_{i=1}^n L(x_i|\theta)$, and we can see that for *e.g.* generalized linear models this will depend on every point $x_1, \ldots, x_n$. So we have to pointwise evaluate likelihoods, and the computational cost will scale linearly in n. That is the main cost of MCMC in Big Data case.

In Big Data case, n is super huge. That results in a quite high cost of finishing a MCMC simulation. It is a fact that datasets of more than 100 GB are now fairly common. Since every iteration has to evaluate each data point at least once, the computational cost of computing the posterior distribution for i.i.d data scales at least linearly with respect to data sizes. The running time of basic MH algorithm or Gibbs algorithm on such a large dataset is prohibitively long.

2.5.2 Austerity MCMC

To cut down computational cost of data sets with product-form likelihoods, some papers naturally consider to use only partial data to approximate the likelihood $L(X|\theta)$ at one step. For example, Korattikara et al (2013) [11] developed an approximate MCMC which used only a random subset of the data at each step to estimate the acceptance probability of the MH algorithm. They use a hypothesis test by comparing the difference of log-likelihoods of partial data to decide if accept a new sample. This action replaces acceptance probability of MH algorithm. If the size of subsets was large enough, the bias can reach a prescribed confidence level. At the same time, the cost of MCMC would also get down to acceptable as compensation of losing accuracy.

2.5.3 Can PMMH Help?

It is quite easy to write down an unbiased estimator of a log-likelihood based on a small sample of a full dataset. This lead many authors to think that some variant of the PMMH algorithm could be used to construct efficient subsampling algorithms. However, the work of Jacob et al [9] showed that this is not possible. They did this by showing the following very general result: given a sequence of unbiased estimators of a given parameter λ and a nonnegative nonconstant function f of interest, there is no generic way to transform this sequence into a nonnegative unbiased estimator of $f(\lambda)$. In particular, there is no way to transform a sequence of unbiased estimators of the log-likelihood into an unbiased estimator of the likelihood.

2.6 Taylor's Expansion

A Taylor series is a representation of a real or complex-valued function $f(x)$ as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point a. It can be written as below

$$f(x) = f(a) + \frac{f^{(1)}(a)}{1!}(x-a) + \frac{f^{(2)}(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots$$

. where $f^{(n)}(a)$ is the nth derivative of f evaluated at point a and $n!$ denotes the factorial of n. The concept of a Taylor series was formulated by the Scottish mathematician James Gregory and formally introduced by the English mathematician Brook Taylor in 1715.

$f(x)$ can also be approximated by using a finite number of terms of its Taylor series. Taylor's theorem gives quantitative estimates on the error introduced by the use of such an approximation. The polynomial formed by taking k initial terms of the Taylor series is called a Taylor polynomial. The rest sums to a remainder term which goes faster to zero

than any nonzero kth degree polynomial as $x \to a$. Taylor's Expansion is to represent f by a Taylor polynomial.

Taylor's Expansion is a common tool to approximate a complicated function with a simple one. In the context of statistical work, this is often used to develop "Laplace approximations" to distributions or to improve the accuracy of simple estimators. Quiroz et al [14] used this to improve their simple sub-sampling PMMH algorithm.

2.7 Convex hull case

It is a statistics problem to compute the cumulative probability. One way to do that is to compute the volume of their distributions. Many probability distributions have a convex hull shape or their shape is able to transformed into a convex hull by some tricks like doing log-concave targets. But to achieve this goal requires more than knowledge in statistics.

In computer science, computing the volume of a convex hull is difficult. There is a large literature about it. In 1987, Bárány et al [2] showed the ratio of the estimated upper and lower bound of the volume of a certain convex hull is quite big. It is not possible to approximate the volume well via deterministic methods. On the other hand, even though we have feasible algorithms to deal with low dimension cases, reduction for a higher dimension is also a big problem. We have to decompose the convex hull by reducing dimension one by one until we can apply on our feasible algorithms. During decomposing, it is not easy to properly separate off the convex hull, either. Simonovits et al [17] pointed out one provably-good random approximate solution was to do a uniform sampling on the hull. Fast-mixing Markov chains are basically the only way to actually do volume computations for such class of problems. Some random works mixes rapidly starting from any interior point of a convex hull. For example, Smith and Robert in 1984 [18] firstly proved hit-and-run walk could produce a uniform stationary distribution over a convex hull. Hence, it is worthwhile to find a good MCMC method of targeting a uniform distribution on a convex hull in Big Data case.

Chapter 3

Algorithm

In this chapter, we start to build our own subsampling MCMC algorithms. Section 3.1 gives some basic notation that is frequently mentioned in Chapter 3 and Chapter 4. We present a naive data-augmentation algorithm similar to FLYMC algorithm in Section 3.2. Since this algorithm is not yet implementable in practice, in Section 3.3 we then add one more variable than FLYMC algorithm involved in computing acceptance probability. We also apply pseudo-marginal theory on the extra variable with a biased estimator to simplify calculation.

3.1 Basic Notation

We fix a dataset $X = (x_1, \ldots, x_n) \in \mathbb{R}^d$. X is parameterized by θ which uniformly distributes on the convex hull composed by X, denoted by $\mathrm{Conv}(X)$. The convex hull of a set $S = \{s_1, \ldots, s_m\} \subset \mathbb{R}^d$ is given by the formula

$$\mathrm{Conv}(S) = \{\sum_{i=1}^{m} \alpha_i s_i \ : \ \forall\, 1 \leq i \leq m, \alpha_i \geq 0, \sum_{i=1}^{m} \alpha_i = 1\}.$$

Define the augmented state space $\Omega = \mathrm{Conv}(X) \times 2^X$. Denote by λ_1 the usual Lebesgue measure on $\mathrm{Conv}(X)$ and λ_2 the usual counting measure on 2^X; all densities on Ω will be with respect to the product measure $\lambda = \lambda_1 \otimes \lambda_2$ on Ω. Also define the collection of size-m subsets $\mathcal{G}_m = \{S \subseteq X : |S| = m\}$ of X. For $\theta \in \mathrm{Conv}(X)$, define

$$\mathcal{G}_m(\theta) = \{S \in \mathcal{G}_m \ : \ \theta \in \mathrm{Conv}(S)\}$$

to be the collection of size-m subsets of X whose convex hulls contain θ. In principle one could replace $\mathcal{G}_m$ with another collection of subsamples with the property that the union of the convex hulls covers $\mathrm{Conv}(X)$; we use $\mathcal{G}_m$ as the simplest family of collections guaranteed to have this property.

3.2 An Initial Algorithm

We define here an initial "idealized" data augmentation Algorithm 7. Like FLYMC algorithm. We also add an auxiliary variable $S \in \mathcal{G}_m$ to mark a subset of the dataset and force the update of θ only depends on S. The cost of drawing θ in every iteration obviously rises up with m increasing. For this reason, we are generally interested in some subsets with a small size. Throughout the paper we will usually be interested primarily in the set with $m = d + 2$ (choosing $m < d + 1$ generally results in a chain that is not ergodic; choosing $m = d + 1$ generally results in a chain that is ergodic, but in practice has some important bottlenecks that prevent rapid mixing).

We next consider a distribution π on our augmented state space Ω. We insist that a sample $(\theta, S) \sim \pi$ drawn from the distribution has marginal distribution equal to the target of interest:

$$\mathbb{P}[\theta \in \cdot] = \text{Unif}[\text{Conv}(X)]. \tag{3.2.1}$$

We denote two conditional probabilities of π by $\pi_\theta(\cdot) = \mathbb{P}[\theta \in \cdot|\theta]$ and $\pi_S(\cdot) = \mathbb{P}[\cdot \in \text{Conv}(S)|S]$. For now, we don't set concrete conditional probabilities.

Following the notation of Theorem 2.3.1, let $K_{G,S} = \pi_\theta(\cdot|\theta, X)$ and $K_{G,\theta} = \pi_S(\cdot|S, X)$. Then $K_{G,S}$ and $K_{G,\theta}$ are the two parts of a Gibbs sampler on our augmented state space. Algorithm 7 listed below targets marginal distribution $\pi_\theta(\theta)$. Algorithm 7 is a case of Algorithm 5.

Algorithm 7 Naive Gibbs Sampler

Initialize $\theta_1 \in \text{Conv}(X)$, $S_1 \in \mathcal{G}_m$ satisfying $\theta_1 \in \text{Conv}(S_1)$.
for $t = 1$ to T **do**
  Sample $\theta_{t+1} \sim \pi_S(\cdot|S_t, X)$.
  Sample $S_{t+1} \sim \pi_\theta(\cdot|\theta_{t+1}, X)$.
end for

Until now Algorithm 7 only works in theory, since we have not assigned a special joint density for $\pi(\theta, S)$ yet. Algorithm 7 can not be called a "real" algorithm either. Next step is to specify such a density ρ by setting its marginal or conditional densities.

3.3 Practicable density on θ

For $(\theta, S) \sim \pi$, denote by ρ the density of (θ, S) with respect to λ, ρ_1 (respectively ρ_2) the density of the marginal distributions of θ (respectively S), and ρ_θ (respectively ρ_S)

the density of the conditional distributions of S given θ (respectively θ given S). We have already insisted in 3.2.1 that the first marginal distribution is

$$\rho_1(\theta) = \frac{1}{Vol(\text{Conv}(X))}, \tag{3.3.1}$$

where $Vol(\cdot)$ is a function to compute a convex hull's volume.

We have a quite wide range of choices on expressions of the conditional distribution of S given θ. Among them, one would be better if its conditional density $\rho_\theta(\cdot)$ is easy to implementable. The simplest choice is the uniform density

$$\rho_\theta(\cdot) \propto \frac{1}{|G_m(\theta)|} \tag{3.3.2}$$

Combining density 3.3.1 and 3.3.2, we can obtain the corresponding joint density:

$$\rho(\theta, S) = \frac{1}{Vol(X)} \frac{1}{|(\mathcal{G}_m(\theta))|} \mathbb{I}(\theta \in \text{Conv}(S)), \tag{3.3.3}$$

where $\mathbb{I}(\cdot)$ is an indicator function which has following probability density function:

$$\mathbb{I}(\cdot) = \begin{cases} 0 & \theta \notin \text{Conv}(S) \\ 1 & \theta \in \text{Conv}(S) \end{cases}$$

It has following corresponding conditional densities:

$$\rho_\theta(S) = \frac{1}{|(\mathcal{G}_m(\theta))|} \mathbb{I}(\theta \in \text{Conv}(S))$$
$$\rho_S(\theta) = \frac{1}{|\mathcal{G}_m(\theta)| \int_{\zeta \in \text{Conv}(S)} \frac{1}{|\mathcal{G}_m(\zeta)|} d\zeta} \mathbb{I}(\theta \in \text{Conv}(S))$$

It is obvious that $\rho_S(\theta)$ is incredibly hard to sample from since the integral in the denominator is computationally intractable. To solely target such a density, we usually build an MH kernel based on a proposal kernel (denoted by $K_S(\cdot)$) that has the uniform distribution on $\text{Conv}(S)$ as its stationary measure. In the simplest case, we set $K_S(\cdot) = \frac{1}{Vol(\text{Conv}(S))}$ as a proposal distribution. In practice we use the hit-and-run algorithm if $Vol(\text{Conv}(S))$ has a big proportion of the whole convex hull.

In any case, we then denote by $K_{MH,S}$ the MH kernel with proposal distribution K_S and target distribution ρ_S. Similarly, we denote by $K_{MH,\theta}$ a kernel with stationary distribution ρ_θ. Together, these form a Metropolis-within-Gibbs algorithm, and by Theorem 2.3.3 the stationary density of this algorithm is 3.3.3. In Algorithm 8, we write down the special case of this algorithm that forms the basis of this thesis:

Algorithm 8 First Naive Alternative Algorithm

Initialize $\theta_1 \in \text{Conv}(X)$ $S_1 \in \mathcal{G}$ satisfying $\theta_1 \in \text{Conv}(S_1)$, and $w_1 = |\mathcal{G}_m(\theta_1)|$.
for $t = 1$ to T **do**
  Propose θ^* from the distribution $K_{S_t}(\cdot) = \frac{1}{Vol(S_t)}$.
  Set $w^* = |\mathcal{G}_m(\theta^*)|$.
  Sample U uniformly in the interval $[0, 1]$.
  if $U \leq \frac{K_{S_t}(\theta^*)w_t}{K_{S_t}(\theta_t)w^*}$ **then**
    Set $\theta_{t+1} = \theta^*$, $w_{t+1} = w^*$.
    Sample S_{t+1} uniformly from the set $\mathcal{G}_m(\theta_{t+1})$.
  else
    Set $\theta_{t+1} = \theta_t$, $w_{t+1} = w_t$ and $S_{t+1} = S_t$.
  end if
end for

However, there is still a difficulty to exactly compute $|\mathcal{G}_m(\theta)|$ by definition, since even a moderately large set X of size *e.g.* $|X| = 200$ can have millions of subsets of size m. Define a new variable $W(\theta) = \frac{|\mathcal{G}_m(\theta)|}{|\mathcal{G}|}$. $W_k(\theta)$ will not change the acceptance probability. But it is easy to estimate $W(\theta)$ by an estimator $W_k(\theta)$ produced by naive Monte Carlo Algorithm 9 below. Its properties are discussed in Section 5.

Algorithm 9 Approx $W_k(\theta)$

Input: point θ^* and order $k \in \mathbb{N}$.
Output: estimator $W_k(\theta^*)$ of of $W(\theta^*)$.
Initialize $N = 0$, $i = 0$.
while $N < k$ **do**
  Sample S^* uniformly from $\mathcal{G}$.
  if $\theta_t \in \text{Conv}(S^*)$ **then**
    set $N = N + 1$.
  end if
  Set $i = i + 1$.
end while
Return the estimator $W_k(\theta^*) = \frac{k}{i}$.

One thing worthwhile to mention is $\frac{1}{W_k(\theta)}$ is the unbiased estimator of $\frac{1}{W(\theta)}$, but $W_k(\theta)$ is not an unbiased estimator of $W(\theta)$. We have two reasons to insist to use $W_k(\theta)$ instead of $\frac{1}{W_k(\theta)}$. One reason is $W_k(\theta)$ is much easier to obtain. Another more important one is by Theorem 2.4.1, bounds on the bias of the individual approximating measure of θ is allowed to be translated into bounds on the error in the stationary measure of the algorithm. So the stationary measure is proportional to the ratio of W_k and W. Then, this new algorithm is Algorithm 10 listed below:

Algorithm 10 Pseudomarginal Approximation of Algorithm 8

Parameter: approximation level $k \in \mathbb{N}$.
Initialize $\theta_1 \in \text{Conv}(X)$, $S_1 \in \mathcal{G}_m(\theta_1)$, and $w_1 > 0$.
for $t = 1$ to T **do**
  Propose θ^* from the distribution $K_{S_t}(\cdot)$.
  Compute $w^* = W_k(\theta^*)$ according to Algorithm 9 by inputting θ^* and k.
  Sample U uniformly in the interval $[0, 1]$.
  if $U \leq \frac{K_{S_t}(\theta^*)w_t}{K_{S_t}(\theta_t)w^*}$ **then**
    Set $\theta = \theta^*$ and $w_{t+1} = w^*$.
    Sample $S_{t+1} \sim \text{Unif}[\mathcal{G}_m(\theta_{t+1})]$.
  else
    Set $\theta_{t+1} = \theta_t$, $w_{t+1} = w_t$ and $S_{t+1} = S_t$.
  end if
end for

Note that this is essentially just Algorithm 8 with W replaced by W_k. By Theorem 2.4.1, the marginal density $\rho^{(m)}$ on $\text{Conv}(X) \times \mathcal{G}_m$ of the stationary measure of Algorithm 10 is:

$$\rho^{(k)}(\theta, S) \propto \mathbb{E}[\frac{W_k(\theta)}{W(\theta)}]\rho(\theta, S).$$

We analyze the bounds of errors in Section 5.

Chapter 4

Advanced Algorithm

In this chapter, we develop two advanced algorithms based on Algorithm 10 in Chapter 3 and corresponding sub-algorithms to achieve them. More spectacular, two powerful tools, Location Sensitive Function (LSH) and Taylor's expansion, are introduced for the purposes of rising up efficiency and accuracy respectively.

4.1 Alternative Proposal Distributions

In algorithm 10, when θ goes into the edge region of the convex hull, $|\mathcal{G}_m(\theta)|$ becomes much smaller. $W_k(\theta)$ accordingly becomes very small even not 0. Then it is more likely to reject a new sample. Always rejecting new samples may affect the algorithm's efficiency. Since we already know the location information (coordinate) of samples θ, why not to use it to select suitable S which contains θ? To do this, we need a certain classifier on data points to determine suitable endpoints before sampling S. Now we introduce Locality Sensitive Hash function (LSH)[7] to propose S.

4.1.1 Build a LSH

LSH is composed by two steps. In the first step, we determine a set of adjusted weights which corresponds to a sample θ. To achieve it, we use cosine similarity to measure two points' original similarity. Under cosine similarity, two points would be similar if the cosine value of an angle between two vector corresponding to points is closer to 1. The measure makes sense because they are more likely to stay in the same tiny small convex hull in this case.

Figure 4.1 gives an example on how to assign weights. *Black Dot* is the original point enclosed by a convex hull X whose endpoints are colorful. *Red Plus* is an arbitrary point (θ). *Black point* and *Red Plus* together decide the *Black Line* crossing them. In Algorithm

11, similarity could be ranked by 9 levels. (9 is a changeable value for other practical cases. We take 9 as an example in this thesis.) Then, the whole circle area is equally separated into 9 zones with different weights. The weight (or similarity) of zones is determined by endpoints lying in it. A zone will have a heavier weight If the smallest angle between endpoints in it and *Black Line* is smaller. In Figure 4.1, different colors stands for different weights of every zone. The explicit formula for computing weights is listed in Algorithm 11.

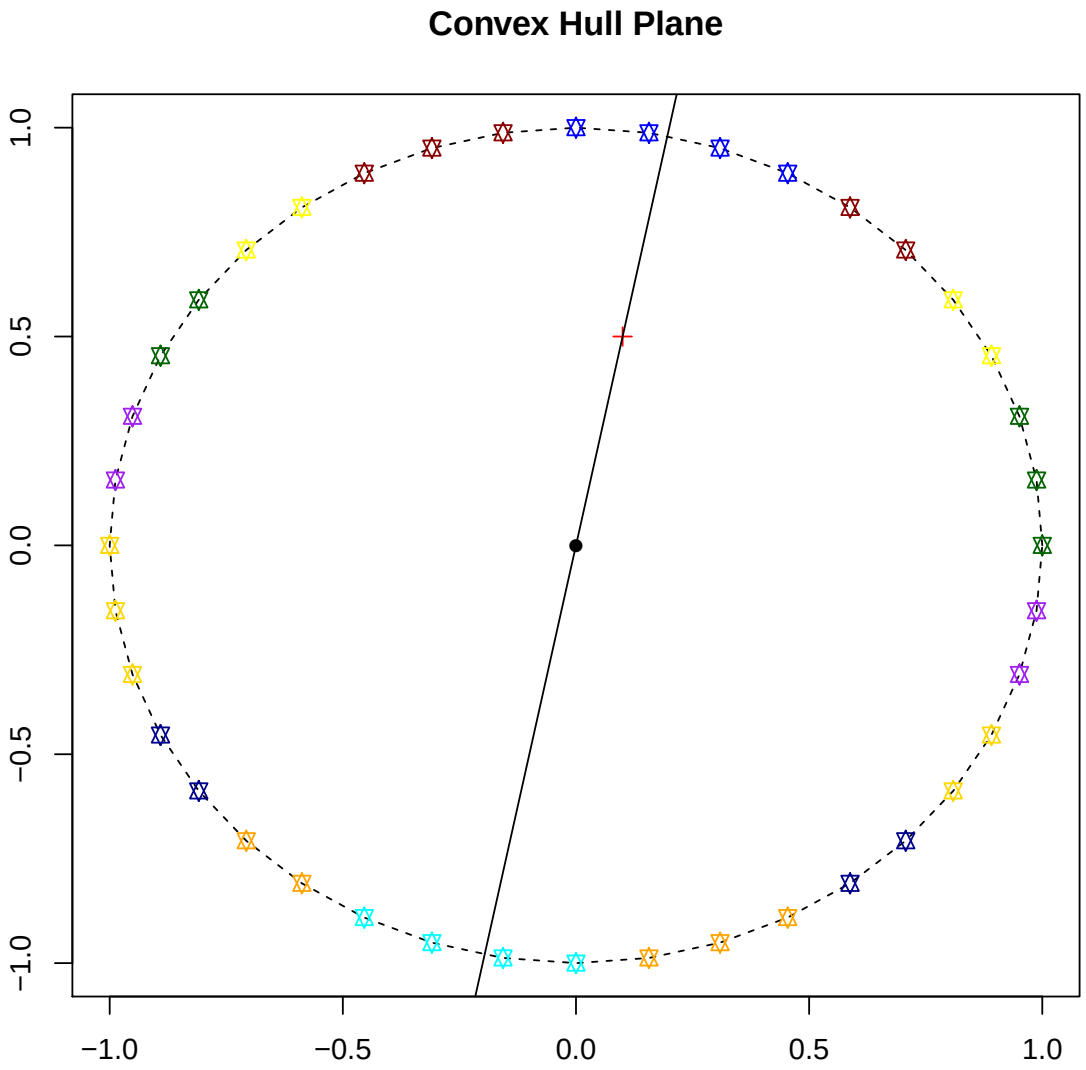

Figure 4.1: How to assign weights

Algorithm 11 Determining weights

Input: $\theta \in \mathrm{Conv}(X)$.
Output: $\{\mathbb{W}_j\}_{j=1}^n$.
Let O be the centre of data set X.
for $j = 1$ to n **do**
Denote by $\mathcal{A}_i \in [0, \pi)$ the angle between the line going from O to θ and the line going from O to x_j.
Define the weight $\mathbb{W}_j = 9 - [\frac{\mathcal{A}_j}{\frac{\pi}{9}}]$.
end for
Return $\{\mathbb{W}_j\}_{j=1}^n$

In Algorithm 11, $[\cdot]$ is an integer-valued function. To make Algorithm 11 executable for all cases, there are two supplementary definitions. When $\mathcal{A}_j = \pi$, $\mathbb{W}_j = 1$ instead of 0. When $\theta = O$, give every endpoint x_j with equal weight $\mathbb{W}_j = 1$ directly and omit the procession of composing an angle.

In the second step, we input adjusted weights from Algorithm 11 to draw S^*. For $X \in \mathbb{R}^2$, we need draw four different endpoints to compose $S^* \in \mathcal{G}_4$. To estimate $W_k(\theta)$, we also need to know the tries of sampling an available S^*. For convenience, we achieve the two goals in Algorithm 12 at the same time.

Algorithm 12 Proposal sampling using LSH with density 4.1.1

Input: θ and $\{\mathbb{W}_j\}_{j=1}^n$.
Output: S^*.
Initialize $Num = 0$.
while STOP=0 **do**
Set $Num = Num + 1$
Sample a set denoted by S^* which contains 4 different endpoints denoted by $\{s_1, s_2, s_3, s_4\}$ from data set X with weights $\{\mathbb{W}_j\}_{j=1}^n$ and without replacement.
if $\mathrm{Conv}(S^*)$ contains θ **then**
STOP=1
else
STOP=0
end if
end while
Return S^* and Num.

Algorithm 12 is a sampling without replacement given a set of weights. The proposal density corresponding to Algorithm 11 and 12 is:

$$P_{LSH}(S^* = \{s_1, s_2, s_3, s_4\}|\theta_t) \propto \tag{4.1.1}$$

$$\sum_{(j1,j2,j3,j4)\in \#A} \frac{\mathbb{W}_{j1}}{W_S} \frac{\mathbb{W}_{j2}}{W_S-\mathbb{W}_{j1}} \frac{\mathbb{W}_{j3}}{W_S-\mathbb{W}_{j1}-\mathbb{W}_{j2}} \frac{\mathbb{W}_{j4}}{W_S-\mathbb{W}_{j1}-\mathbb{W}_{j2}-\mathbb{W}_{j3}}$$

where $\#A$ is a set containing all permutation of $(j1, j2, j3, j4)$.

4.1.2 Adjust $W_k(\theta)$

Because LSH gives endpoints different weights, we must make corresponding adjustment to the manner of calculating $W_k(\theta)$ when counting eligible θ. Since Algorithm 9 involves drawing S^* too, the sub-algorithm 9 becomes Algorithm 13 below:

Algorithm 13 Adjusted $W_k(\theta)$ with weights

Input: point θ, order $k \in \mathbb{N}$.
Output: estimator $W_k(\theta)$ of $W(\theta)$.
Initialize $N = 0$, $i = 0$.
Input θ into Algorithm 11 and get a set of weights $\{\mathbb{W}_j\}_{j=1}^n$.
while $N < k$ **do**
  Input θ and $\{\mathbb{W}_j\}_{j=1}^n$ into Algorithm 12 and get a subset S^* as well as tries Num
  Set $i = i + Num$.
end while
Return the estimator $W_k(\theta) = \frac{k}{i}$.

Note that $W_k(\theta)$ produced by Algorithm 13 and 9 are not the same value. The values of $W_k(\theta)$ produced by Algorithm 13 obviously are bigger than the values produced by Algorithm 9. Algorithm 13 needs fewer tries to get as many S^* containing θ as Algorithm 9 because of the aids by $\{\mathbb{W}_j\}_{j=1}^n$.

Since Algorithm 13 adjusts probability of every S of $\mathcal{G}_m(\theta)$, denote a new function $\#(S,\theta)$ which satisfies $Pr(S|\theta) = \frac{\#(S,\theta)}{\sum_{\xi\in\mathcal{G}_m(\theta)}\#(\xi,\theta)}$, where $Pr(S)$ is the probability of getting a S from the collection $\mathcal{G}_m(\theta)$. Once we know the location of θ, $Pr(S|\theta)$ is decided because of $P_{LSH}(S|\theta)$. For convenience, denote $\#_\theta = \sum_{\xi\in\mathcal{G}_m(\theta)}\#(\xi,\theta)$ and denote $\# = \int_{\theta\in\mathrm{Conv}(X)}\#_\theta d\theta$. Then Algorithm 13 produces $\mathbb{E}[W_k(\theta)] = \frac{\#_\theta}{\#}$. With those modified and additional definitions, joint density 3.3.3 becomes density 4.1.2

$$\rho(\theta, S) = \frac{1}{Vol(X)} \frac{\#(S,\theta)}{\#_\theta} \mathbb{I}(\theta \in \mathrm{Conv}(S)). \tag{4.1.2}$$

The acceptance probability of θ in step t for density 4.1.2 is

$$AcceptanceProbability = \frac{K_{S_t}(\theta_t,\theta^*)\frac{1}{\#_{\theta^*}\int_{\zeta\in\mathrm{Conv}(S_t)}\frac{1}{\#_\zeta}d\zeta}}{K_{S_t}(\theta^*,\theta_t)\frac{1}{\#_{\theta_t}\int_{\zeta\in\mathrm{Conv}(S_t)}\frac{1}{\#_\zeta}d\zeta}} = \frac{\#_{\theta_t}}{\#_{\theta^*}}.$$

If we divide # for both numerator and denominator, the acceptance probability still equals to $\frac{W_k(\theta_t)}{W_k(\theta^*)}$. So, no matter which $W_k(\theta)$ produced by Algorithm 13 or 9 will both work for their own algorithms.

Algorithm 10 only uses one MH kernel $K_{MH,\theta}$. By Theorem 2.3.3, it is safe to replace $K_{G,S}$ with a MH kernel $K_{MH,S}$. That is, we are able to use P_{LSH} as a proposal distribution for S to increasing efficiency. A new algorithm (Algorithm 14) adding LSH in algorithm 10 is listed below. Its density is Density (4.1.2).

Algorithm 14 Algorithm 10 equipped with LSH

Parameter: approximation level $k \in \mathbb{N}$.
Initialize $\theta_1 \in \text{Conv}(X)$.
Determine endpoints' weights $\{\mathbb{W}_j\}_{j=1}^n$ by inputting θ_1 to Algorithm 11
Draw S_1 by inputting θ_1 and $\{\mathbb{W}_j\}_{j=1}^n$ to Algorithm 12
Calculating w_1 according to Algorithm 13
for $t = 1$ to T **do**
  Propose θ^* from the distribution $K_{S_t}(\cdot)$.
  Compute $w^* = W_k(\theta^*)$ according to Algorithm 13 by inputting θ^*, k and $\{\mathbb{W}_j\}_{j=1}^n$.
  Sample U uniformly in the interval $[0, 1]$.
  if $U \leq \frac{K_{S_t}(\theta^*)w_t}{K_{S_t}(\theta_t)w^*}$ **then**
    Determine new set of weights $\{\mathbb{W}_j^*\}_{j=1}^n$ according to Algorithm 11 by inputting θ_t
    Propose S^* according to Algorithm 12 by inputting θ_t and $\{\mathbb{W}_j^*\}_{j=1}^n$.
    Sample V uniformly in the interval $[0, 1]$.
    if $V \leq \frac{P_{LSH}(S^*|\theta_t)}{P_{LSH}(S_{t|\theta_t})}$ **then**
      Set $\theta_{t+1} = \theta^*$, $w_{t+1} = w^*$, $S_{t+1} = S^*$ and substitute $\{\mathbb{W}_j\}_{j=1}^n$ with $\{\mathbb{W}_j^*\}_{j=1}^n$.
    else
      Set $\theta_{t+1} = \theta_t$, $w_{t+1} = w_t$, $S_{t+1} = S_t$ and keep $\{\mathbb{W}_j\}_{j=1}^n$.
    end if
  else
    Set $\theta_{t+1} = \theta_t$, $w_{t+1} = w_t$, $S_{t+1} = S_t$ and keep $\{\mathbb{W}_j\}_{j=1}^n$.
  end if
end for

4.2 Taylor's expansion of $W(\theta)$

According to algorithm 9[1], we define the estimator $W_k(\theta) = \frac{k}{i}$ for $W(\theta)$, where k is a given number counting repeating samples and i is the total number of Bernoulli experiments under negative binomial distribution. For a fixed point θ, i is also a random variable

[1]Since adjusted weight also applies on counting $|\mathcal{G}_m(\theta)|$ and $|\mathcal{G}|$, new definition of $\mathcal{G}'$ and $\mathcal{G}'_m$ does not affect following properties. They still work in Algorithm 14.

with density function $f(i) = p^k(1-p)^{i-k}\frac{(i-1)!}{(i-k)!(k-1)!}$, where the probability of successful experiments $p = W(\theta)$. The expectation $\mathbb{E}[i]$ and variance $\text{Var}[i]$ of the random variable are $\frac{k}{p}$ and $k\frac{1-p}{p^2}$ respectively.

We note that $\frac{1}{W_k(\theta)}$ is an unbiased estimator of $\frac{1}{W(\theta)}$, but $\mathbb{E}[W_k(\theta)] \neq W(\theta)$. Instead, we merely have $lim_{k\to\infty}W_k(\theta) = W(\theta)$. The goal of this section is to reduce (but not eliminate) this bias by replacing our estimator $W_k(\theta)$ with a slightly perturbed estimator.

For $h^{-1} = k \in \mathbb{N}$, define a new function $g_{P(\theta)}(h) = \mathbb{E}[W_k(\theta)] = \mathbb{E}[\frac{k}{i}]$, where subscript $P(\theta) = \{\theta' \in \Theta : W(\theta') = W(\theta)\}$. Particularly, $g_{P(\theta)}(0) = lim_{k\to\infty}W_k(\theta) = W(\theta) = p$. Since we are interested in $W(\theta)$, we can use a Taylor expansion of g around 0 to increase the precision of our estimator.

We claim that a first-order Taylor expansion yields:

$$g_{P(\theta)}(0) = \frac{W_k(\theta) - \frac{1}{k}}{1 - \frac{1}{k}} + \delta(h) \tag{4.2.1}$$

where $\delta(h)$ is a error term relative to h and has $lim_{h\to 0}\frac{\delta(h)}{g_{P(\theta)}(0)-\delta(h)} = 0$.

Proof of Equation (4.2.1). Firstly, apply Taylor expansion on $g_\theta(h)$. That is

$$g_{P(\theta)}(0) = g_{P(\theta)}(h) - h \cdot g'_{P(\theta)}(h) + R_2(h) \tag{4.2.2}$$

Here $R_2(h)$ is a reminder term of the Taylor expansion and it has $lim_{h\to 0}\frac{R_2(h)}{h} = 0$.

Secondly, we try to find the expression of $g'_{P(\theta)}(h)$. Define a random variable $c = i - \mathbb{E}[i]$. Its expectation $\mathbb{E}[c]$ and second moment $\mathbb{E}[c^2]$ are $\mathbb{E}[c] = 0$ and $\mathbb{E}[c^2] = Var[i] = k\frac{1-p}{p^2}$ respectively. Then $\mathbb{E}[W_k(\theta)] = \mathbb{E}[\frac{k}{i}] = \mathbb{E}[\frac{k}{\mathbb{E}[i]+c}] = \mathbb{E}[\frac{kp}{k+cp}]$. Denote $\frac{kp}{k+cp}$ as function $f(c) = \frac{kp}{k+cp}$. Apply Taylor's expansion theory on Function $f(c)$ at point $c = 0$. Then we have $f(c) = p - \frac{p^2}{k}c + \frac{p^3}{k^2}c^2 + \frac{f^3(H)}{3!}c^3$, where $\frac{f^3(H)}{3!}c^3$ (denoted by $R_3(c)$) is a remainder term that also has $lim_{c\to 0}\frac{R_3(c)}{c^2} = 0$ and $H \in [\frac{k(p-1)}{p}, \infty)$. Therefore, apply Taylor's expansion theory on $g_{P(\theta)}(0)$ again

$$\begin{aligned} g_{P(\theta)}(0) = \mathbb{E}[\frac{k}{i}] &= \mathbb{E}[p - \frac{p^2}{k}c + \frac{p^3}{k^2}c^2 + R_3(H)] \\ &= p - \frac{p^2}{k}\mathbb{E}[c] + \frac{p^3}{k^2}\mathbb{E}[c^2] + \mathbb{E}[R_3(H)] \\ &= p + \frac{1-p}{k} + \mathbb{E}[R_3(c)] \end{aligned}$$

After that, take the derivation $\frac{\partial(p+\frac{1-p}{k}+\mathbb{E}[R_3(c)])}{\partial h}$. Since $\frac{\partial\mathbb{E}[R_3(c)]}{\partial h} = 0$, then, we get

$$g'_{P(\theta)}(h) = 1 - p \tag{4.2.3}$$

Finally, use Equation 4.2.3 to replace the term $g'_{P(\theta_t)}(h)$ in Equation 4.2.2. We have $g_{P(\theta)}(0) = g_{P(\theta)}(h) - h \cdot g'_{P(\theta)}(h) + R_2(h)$. Additionally, we assume $g_{P(\theta)}(h)$ can also be roughly estimated by $W_k(\theta)$ and $R_2(h)$ would be infinitely close to 0. Finally, we get $g_{P(\theta)}(0) = W_k(\theta) - h(1-p) + R_2(h)$. That is

$$\begin{aligned} g_{P(\theta)}(0) &= \frac{W_k(\theta) - \frac{1}{k}}{1 - \frac{1}{k}} + \frac{R_2(h)}{1-h} \\ &= \frac{W_k(\theta) - \frac{1}{k}}{1 - \frac{1}{k}} + \delta(h) \end{aligned}$$

Obviously, $lim_{h\to 0}\delta(h) = lim_{h\to 0}\frac{R_2(h)}{1-h} = 0$. And then $lim_{h\to 0}\frac{\delta(h)}{g_{P(\theta)}(0)-\delta(h)} = 0$. □

Above deduction suggests we would better substitute $W_k(\theta_t)$ with $\frac{W_k(\theta_t)-\frac{1}{k}}{1-\frac{1}{k}}$ as an estimator of $W(\theta)$ in previous pseudo-marginal algorithms (Algorithm 10 or 14). Such a substitute will not invalidate kernel $K_{MH,\theta}$ because it is still under the PMMH framework and even performs better. We carry out this improvement in Algorithm 14 and get an advanced version Algorithm (with the density (4.1.2)) is listed below:

Algorithm 15 Advanced Algorithm

Parameter: approximation level $k \in \mathbb{N}$.
Initialize $\theta_1 \in \text{Conv}(X)$.
Determine endpoints' weights $\{\mathbb{W}_j\}_{j=1}^n$ by inputting θ_1 to Algorithm 11
Draw S_1 by inputting θ_1 and $\{\mathbb{W}_j\}_{j=1}^n$ to Algorithm 12
Calculating w_1 according to Algorithm 13
for $t = 1$ to T **do**
  Propose θ^* from the distribution $K_{S_t}(\cdot)$.
  Compute $w^* = W_k(\theta^*)$ according to Algorithm 13 by inputting θ^*, k and $\{wg_i\}_{i=1}^n$ Algorithm 12.
  Compute $w^{**} = \frac{w^* - \frac{1}{k}}{1 - \frac{1}{k}}$
  Sample U uniformly in the interval $[0, 1]$.
  if $U \leq \frac{K_{S_t}(\theta^*, \theta_t) w_t}{K_{S_t}(\theta_t, \theta^*) w^{**}}$ **then**
    Determine new set of weights $\{\mathbb{W}_j^*\}_{j=1}^n$ according to Algorithm 11 by inputting θ_t
    Propose S^* according to Algorithm 12 by inputting θ_t and $\{\mathbb{W}_j^*\}_{j=1}^n$.
    Sample V uniformly in the interval $[0, 1]$.
    if $V \leq \frac{P_{LSH}(S^* | \theta_t)}{P_{LSH}(S_{t|\theta_t})}$ **then**
      Set $\theta_{t+1} = \theta^*$, $w_{t+1} = w^{**}$, $S_{t+1} = S^*$ and substitute $\{\mathbb{W}_j\}_{j=1}^n$ with $\{\mathbb{W}_j^*\}_{j=1}^n$.
    else
      Set $\theta_{t+1} = \theta_t$, $w_{t+1} = w_t$, $S_{t+1} = S_t$ and keep $\{\mathbb{W}_j\}_{j=1}^n$.
    end if
  else
    Set $\theta_{t+1} = \theta_t$, $w_{t+1} = w_t$, $S_{t+1} = S_t$ and keep $\{\mathbb{W}_j\}_{j=1}^n$.
  end if
end for

Chapter 5

Analysis of Algorithm 10

After finishing implementable algorithms, we also need to discuss the bound of the convex hull because it is relative to the bias of stationary distribution. When processing a Markov chain, samples of θ from the central area or edge area of the convex hull will probably make $W_k(\theta)$ take 0 or 1. Such odd values may cause errors. We also worry about if the bias will not converge because of odd values. In this section, we prove a theorem which guarantees the total error on the whole convex hull will converge with k increasing.

5.1 Approaching variable $W(\theta)$

Algorithm 10, Algorithm 13 and Algorithm 15 utilize three different methods to estimate $W(\theta)$. Algorithm 9 is the simplest and most fundamental. The other two both make some modification on Algorithm 9. To simplify the problem, we are going to use the simplest $W_k(\theta)$ from Algorithm 9 to analyze.

Define $\omega_k(\theta) = Z_k^{-1}\mathbb{E}[W_k(\theta)]$ where $Z_k = \int_{\theta\in \mathrm{Conv}(X)} \mathbb{E}[W_k(\theta)]d\theta$ is a normalizing constant. Particularly, since $W(\theta) = lim_{k\to\infty}W_k(\theta)$, $\omega(\theta) = Z_\infty^{-1}W_\infty(\theta)$, where Z_∞ is also a normalizing constant $Z_\infty = \int_{\theta\in \mathrm{Conv}(X)} W(\theta)d\theta$.

By these notations, we can use $|\omega(\theta) - \omega_k(\theta)|$ to show the error between $W_k(\theta)$ and $W(\theta)$ for a certain point. We prove a theorem which guarantees the total error on the whole convex hull will converge with k increasing.

Theorem 5.1.1. $\int_{\mathrm{Conv}(X)} |\omega(\theta) - \omega_k(\theta)|d\theta \le 8\sqrt{\frac{\pi}{k}}$

Before giving a proof, we prove two lemmas. For convenience, denote Area as the volume of *Convex Hull* $\mathrm{Conv}(X)$. $Area = \int_{\theta\in \mathrm{Conv}(X)} d\theta$

Lemma 5.1.2. *For* $\forall\theta \in \mathrm{Conv}(X)$,

$$\mathbb{E}|W_k^{-1}(\theta) - W^{-1}(\theta)| \le \sqrt{\frac{2\omega}{k}}.$$

Proof of Lemma 5.1.2. For $\forall \varepsilon_1 > 0$

$$\mathbb{P}(|W_k^{-1}(\theta) - W^{-1}(\theta)| > \varepsilon_1) = \mathbb{P}(|\frac{i}{k} - \frac{1}{p}| > \varepsilon_1) = \mathbb{P}(i \geq \frac{k}{p}(1+\varepsilon_1 p)) + \mathbb{P}(i \leq \frac{k}{p}(1-\varepsilon_1 p))$$

Daniel [3] put forward a bound on lower tail of a sum of geometric distribution. By his method, we get $\mathbb{P}(i \geq \frac{k}{p}(1+\varepsilon_1 p) \leq exp(-\frac{(\varepsilon_1 p)^2 k}{2(1+\varepsilon_1 p)}))$. Also, in his paper, he mentioned a standard theorem to work out upper tail. Then we have $\mathbb{P}(i \leq \frac{k}{p}(1-\varepsilon_1 p)) \leq exp(-\frac{(\varepsilon_1 p)^2 k}{2p})$.

Thus,

$$\mathbb{P}(|\frac{i}{k} - \frac{1}{p}| > \varepsilon_1) \leq exp(-\frac{(\varepsilon_1 p)^2 k}{2(1+\varepsilon_1 p)}) + exp(-\frac{(\varepsilon_1 p)^2 k}{2p}) \leq 2exp(-\frac{(\varepsilon_1 p)^2 k}{2(1+\varepsilon_1 p)}) \quad (5.1.1)$$

Since we always have $0 < \varepsilon_1 p < 1$, inequality 5.1.1 can be magnified to

$$\mathbb{P}(|\frac{i}{k} - \frac{1}{p}| > \varepsilon_1) \leq 2exp(-\frac{(\varepsilon_1 p)^2 k}{4}) \quad (5.1.2)$$

Define $\varepsilon_2 = \varepsilon_1 p(1+p)$ and $0 < \varepsilon_2 < 1$. ε_1 and ε_2 satisfy following inequality group:

$$\begin{cases} \varepsilon_2 - W(\theta) + \frac{1}{W^{-1}(\theta)+\varepsilon_1} > 0 \\ -\varepsilon_2 - W(\theta) + \frac{1}{W^{-1}(\theta)-\varepsilon_1} < 0 \end{cases}$$

Therefore, we have

$$\begin{aligned}
&\{|W_k(\theta) - W(\theta)| > \varepsilon_2\} \\
&= \{W_k(\theta) > W(\theta) + \varepsilon_2\} \cup \{W_k(\theta) < W(\theta) - \varepsilon_2\} \\
&\subseteq \{W_k(\theta) > W(\theta) + \varepsilon_2 - \varepsilon_2 - W(\theta) + \frac{1}{W^{-1}(\theta) - \varepsilon_1}\} \\
&\quad \cup \{W_k(\theta) < W(\theta) - \varepsilon_2 + \varepsilon_2 - W(\theta) + \frac{1}{W^{-1}(\theta) + \varepsilon_1}\} \\
&= \{W_k(\theta) > \frac{1}{W^{-1}(\theta) - \varepsilon_1}\} \cup \{W_k(\theta) < \frac{1}{W^{-1}(\theta) + \varepsilon_1}\} \\
&= \{W_k^{-1}(\theta) < W^{-1}(\theta) - \varepsilon_1\} \cup \{W_k^{-1}(\theta) > W^{-1}(\theta) + \varepsilon_1\} \\
&= \{|W_k^{-1}(\theta) - W^{-1}(\theta)| > \varepsilon_1\}
\end{aligned}$$

Denoting right side of inequality 5.1.2 as bound function $f(\varepsilon_1)$. In order to bound on $\mathbb{P}(|W_k(\theta) - W(\theta)| > \varepsilon_2)$, we have

$$\mathbb{P}(|W_k(\theta) - W(\theta)| > \varepsilon_2) \leq \mathbb{P}(|W_k^{-1}(\theta) - W^{-1}(\theta)| > \frac{\varepsilon_2}{p(p+1)}) < f(\frac{\varepsilon_2}{p(p+1)})$$

Since $0 \leq W_k(\theta), W(\theta) \leq 1$, $0 \leq |W_k(\theta) - W(\theta)| \leq 1$. Then we can calculate its expectation.

$$\begin{aligned}
\mathbb{E}|W_k(\theta) - W(\theta)| &= \int_0^1 P(|W_k(\theta) - W(\theta)| > \varepsilon_2) d\varepsilon_2 \\
&\leq \int_0^1 2exp(-\frac{k\varepsilon_2^2}{4(p+1)^2}) d\varepsilon_2 \\
&= 2 \cdot \sqrt{2\omega}\sigma \int_0^1 \frac{1}{\sqrt{2\omega}\sigma} exp(-\frac{\varepsilon_2^2}{2\sigma^2}) d\varepsilon_2 \\
&< 4\sqrt{\frac{\omega}{k}}(1+p)(\phi(\frac{1}{\sigma}) - 0.5) \\
&< 4\sqrt{\frac{\omega}{k}}
\end{aligned}$$

where $\sigma = (1+p)\sqrt{\frac{2}{k}}$ and $\phi(x)$ is standard Gaussian PDF

□

Lemma 5.1.3.

$$|Z_k - Z_\infty| < 4\sqrt{\frac{\omega}{k}} Area$$

Proof of Lemma 5.1.3.

$$\begin{aligned}
|Z_k - Z_\infty| &= |\int_{\text{Conv}(X)} \mathbb{E}[W_k(\theta)] - W(\theta) d\theta| \\
&\leq \int_{\text{Conv}(X)} \mathbb{E}|W_k(\theta) - W(\theta)| d\theta \\
&< \int_{\text{Conv}(X)} 4\sqrt{\frac{\omega}{k}} d\theta \qquad (by \quad Lemma 5.1.2) \\
&= 4\sqrt{\frac{\omega}{k}} Area
\end{aligned}$$

□

With above two lemmas, we are able to prove our Theorem 5.1.1

Proof of Theorem 5.1.1.

$$\begin{aligned}
\int_{\mathrm{Conv}(X)} |\omega(\theta) - \omega_k(\theta)| d\theta &= \int_{\mathrm{Conv}(X)} |Z_\infty^{-1} W(\theta) - Z_k^{-1} \mathbb{E}[W_k(\theta)]| d\theta \\
&= \int_{\mathrm{Conv}(X)} |Z_\infty^{-1}(W(\theta) - \mathbb{E}[W_k(\theta)]) + \mathbb{E}[W_k(\theta)](Z_\infty^{-1} - Z_k^{-1})| d\theta \\
&< \int_{\mathrm{Conv}(X)} |Z_\infty^{-1}(W(\theta) - \mathbb{E}[W_k(\theta)])| + |\mathbb{E}[W_k(\theta)](Z_\infty^{-1} - Z_k^{-1})| d\theta \\
&< Z_\infty^{-1} \int_{\mathrm{Conv}(X)} \mathbb{E}|W(\theta) - W_k(\theta)| d\theta + |Z_\infty^{-1} - Z_k^{-1}| \int_{\mathrm{Conv}(X)} \mathbb{E}[W_k(\theta)] d\theta \\
(by \quad Lemma 5.1.2) &< Z_\infty^{-1} \int_{\mathrm{Conv}(X)} 4\sqrt{\frac{\pi}{k}} d\theta + Z_k |Z_\infty^{-1} - Z_k^{-1}| \\
&= Z_\infty^{-1} 4\sqrt{\frac{\pi}{k}} Area + Z_\infty^{-1} |Z_k - Z_\infty| \\
(by \quad Lemma 5.1.3) &< Z_\infty^{-1} 4\sqrt{\frac{\pi}{k}} Area + Z_\infty^{-1} 4\sqrt{\frac{\pi}{k}} Area \\
&= 8 Z_\infty^{-1} \sqrt{\frac{\pi}{k}} Area
\end{aligned}$$

Since $W(\theta) < 1$, $Z_\infty = \int_{\mathrm{Conv}(X)} W(\theta) d\theta$ is smaller than the volume of Convex Hull $\mathrm{Conv}(X)$. Therefore,

$$\int_{\mathrm{Conv}(X)} |\omega(\theta) - \omega_k(\theta)| d\theta < 8 Z_\infty^{-1} \sqrt{\frac{\pi}{k}} Area < 8\sqrt{\frac{\pi}{k}}$$

□

Chapter 6

Simulation

This chapter contains many pictures of different themes to show how well our algorithms perform in simulation. Graphs for various target measures give some evidence that our efforts to improve the basic version of the algorithm are successful.

In this chapter, we begin with building a toy data set that will be used for all of our tests with different themes. We firstly evaluate the accuracy and cost of our estimate $W_k(\theta)$. Then, we study the measure distribution of algorithm 10, algorithm 14 and algorithm 15. Finally, we find a pair of best combo between the number n of Markov Chain steps we take in our algorithm and the parameter k used to compute $W_k(\theta)$ for make best use of computational source in practice.

6.1 Default Setting

We select 40 points with equal distance from a unit circle to compose a toy data set. In section 6.2, we choose 4 different θ lying in typical locations in the unit circle to test different properties of $W_k(\theta)$. In section 6.3, we fix parameter $k = 40$ and iteration number $n = 20000$. Following the default setting, we run algorithm 10, algorithm 14 and algorithm 15 and get many graphs to display performance of all algorithms. In section 6.4, we test more pairs of (k, n) on the default setting.

In the following content, legends will call algorithm 10 "Original version" or "Basic version", call algorithm 14 "LSH version" and call algorithm 15 "Taylor version".

6.2 Evaluation of $W_k(\theta)$

Given a sequence of k from 1 to 500, Figure 6.1 shows the corresponding values of $W_k(\theta)$ obtained by simulation. In each figure, we plot the mean value of $W_k(\theta)$ in red and upper

and lower bounds on $W_k(\theta)$ in black. The upper and lower bounds are lined some selected points where we repeat calculating $W_k(\theta)$'s values. Then we compose a 95% confidence interval by CLT. That is, $upper \quad bound = mean + Z_{0.975} * sd$ and $lower \quad bound = mean - Z_{0.975} * sd$. We note that for large enough k, the upper and lower bounds seem to converge. $W_k(\theta)$ will converge to a bigger value if θ locates in more central region.

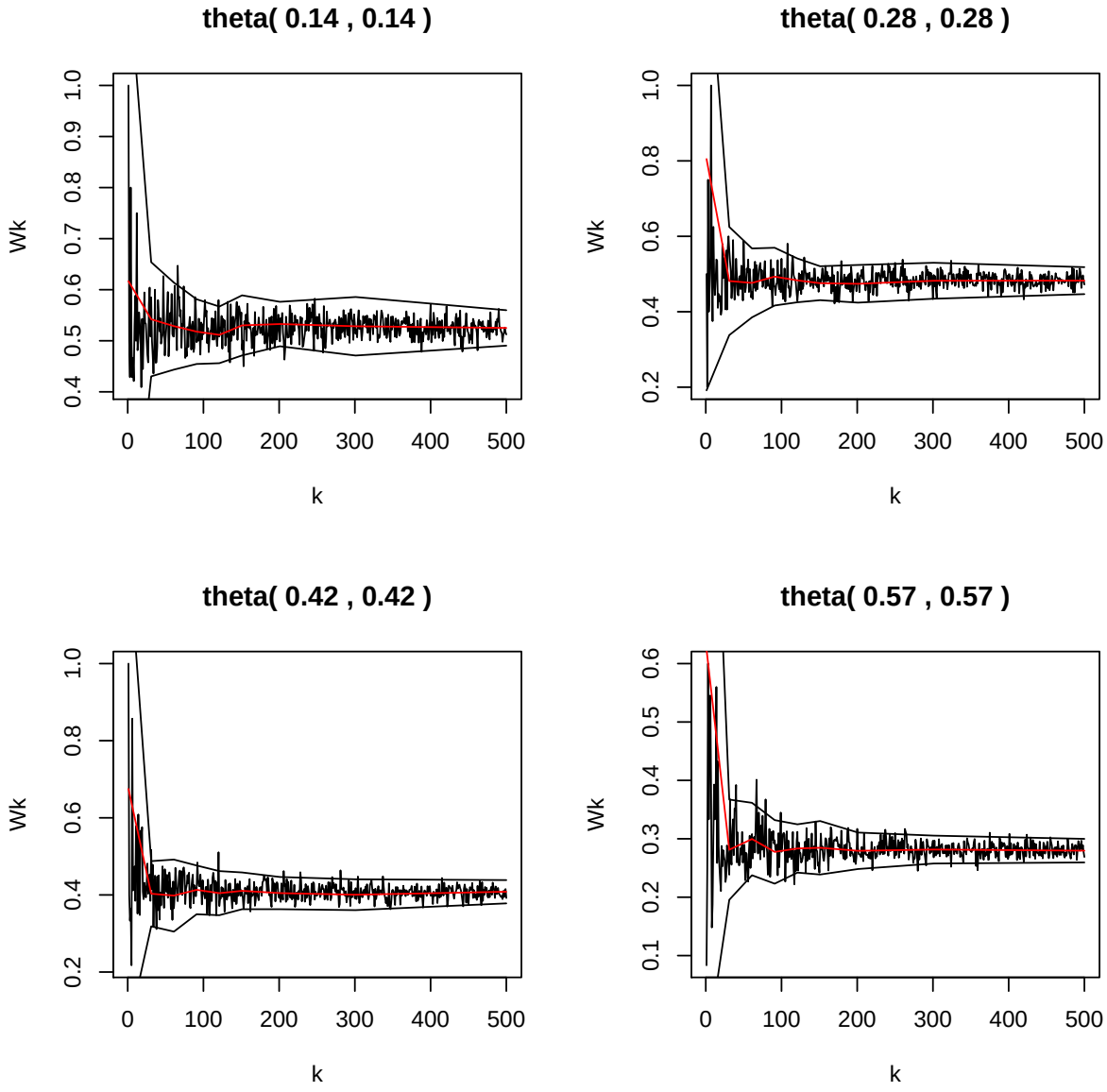

Figure 6.1: Evaluate $W_k(\theta)$ for typical θ

Since the main cost of each iteration comes from the calculation of $W_k(\theta)$, we then count

how many trials is needed to obtain k successful trials. Figure 6.2 shows the relationship between parameter k and calculation number which is the number of trials to obtain k successful trials. With increasing of k, the computation number increases nearly linearly wherever θ lies. In fact, the cost is *exactly* linear in theory. The more important thing is that the cost goes up as θ gets closer to the boundary. When fixing k, the calculation number of bottom right figure is the biggest one because the corresponding θ is farthest from the original point in the circle.

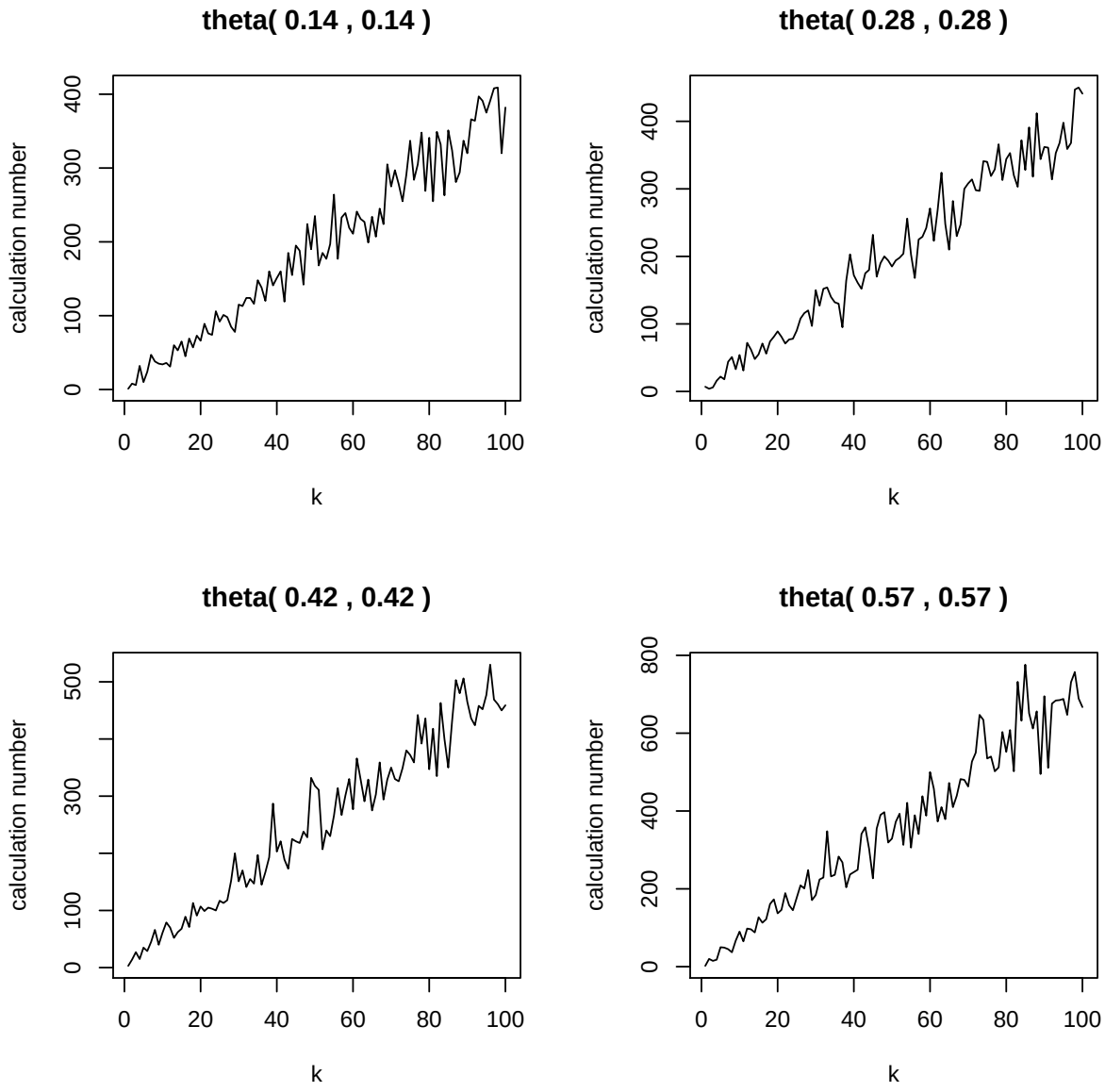

Figure 6.2: Cost of $W_k(\theta)$

6.3 Evaluation of algorithms

In order to visualize the marginal distribution of θ, we plot the point density $D(\|\theta\|)$ with respect to radius $\|\theta\|$, the distance between data points and the original point. The true density in our toy case is as below:

$$D(\|\theta\|) = \begin{cases} \frac{2\pi\|\theta\|}{Area} & 0 \leq \|\theta\| < cos(\frac{\pi}{40}) \\ \frac{2\|\theta\|(\pi - 40arccos(cos(\frac{\pi}{40})))}{Area} & cos(\frac{\pi}{40}) \leq \|\theta\| \leq 1 \end{cases}$$

Where $Area$ is the square of our convex hull X, $Area = 40sin(\frac{\pi}{40})cos(\frac{\pi}{40})$. 40 is the number of endpoints.

Figure 6.3 display the true and estimated densities of the stationary measure of Algorithm 10, Algorithm 14 and Algorithm 15. From the figure, we can confirm that all of the three algorithms are able to produce an approximately uniform marginal density. All lines of three versions are close to true density. Algorithm 14 and Algorithm 15 are less flat than Algorithm 10 because within the same iteration number Algorithm 14 and Algorithm 15 actually cut down much time consumption by skipping many points. If continually extending Markov chains, density lines will be more flat.

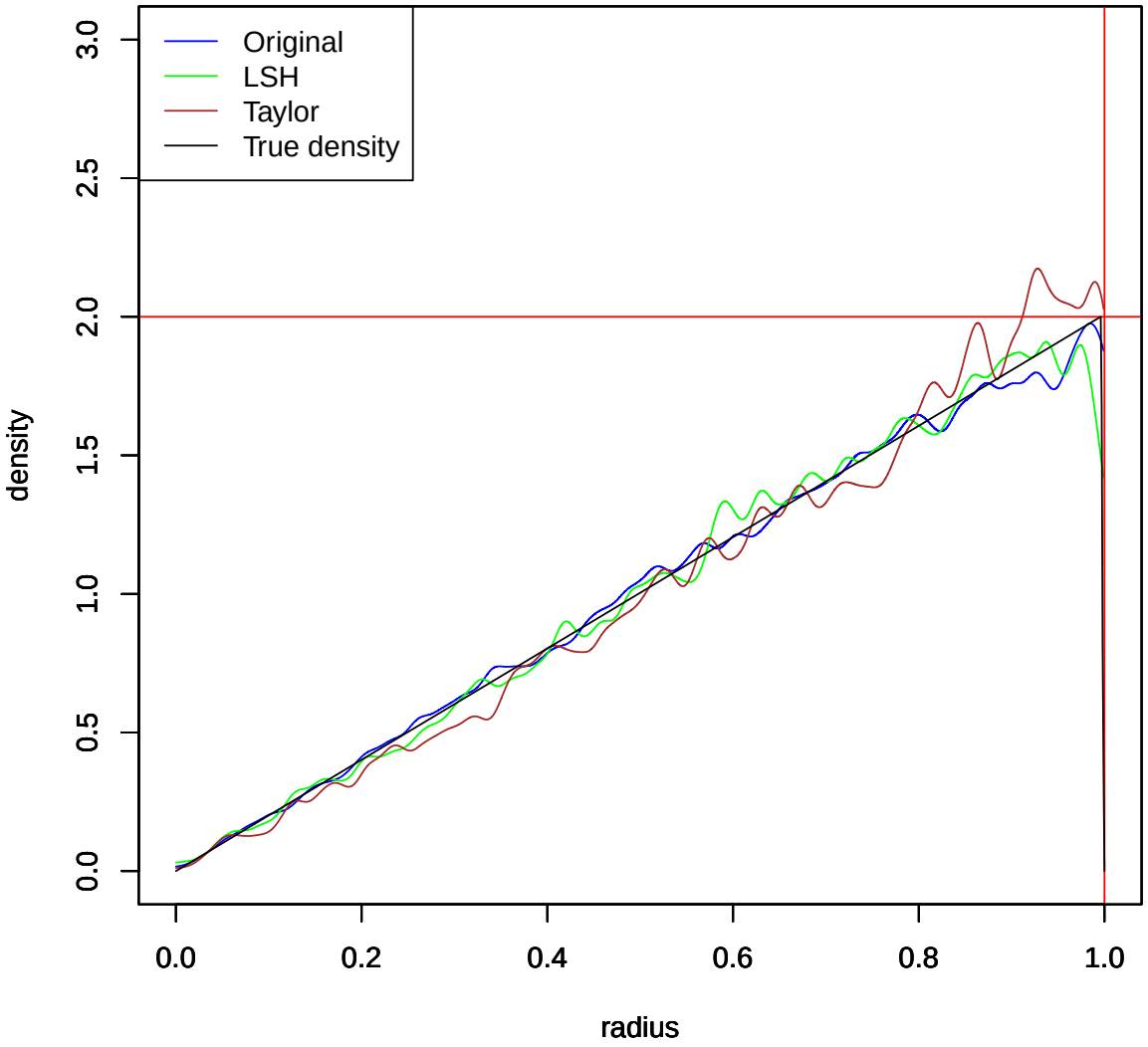

Figure 6.3: density

Next three spaghetti plots trace Mean-squares errors (MSE) of Markov Chains for three versions when chains goes longer. MSE is a sum of a "bias" and "variance" term. Since our algorithm is not an unbiased estimator, MSE will converge to a not-zero value. For the three algorithms, we respectively observe 5 chains (blue lines) with a fixed parameter $k = 40$. Red bounds present a 95% confidence interval by CLT. MSE of the three algorithms quickly reduce to a low and acceptable level even though there are still a fluctuation range. What is worth mentioning is Taylor version has a much more narrower fluctuation range

in the end. Therefore, our modification (algorithm 13) on calculating $W_k(\theta)$ works.

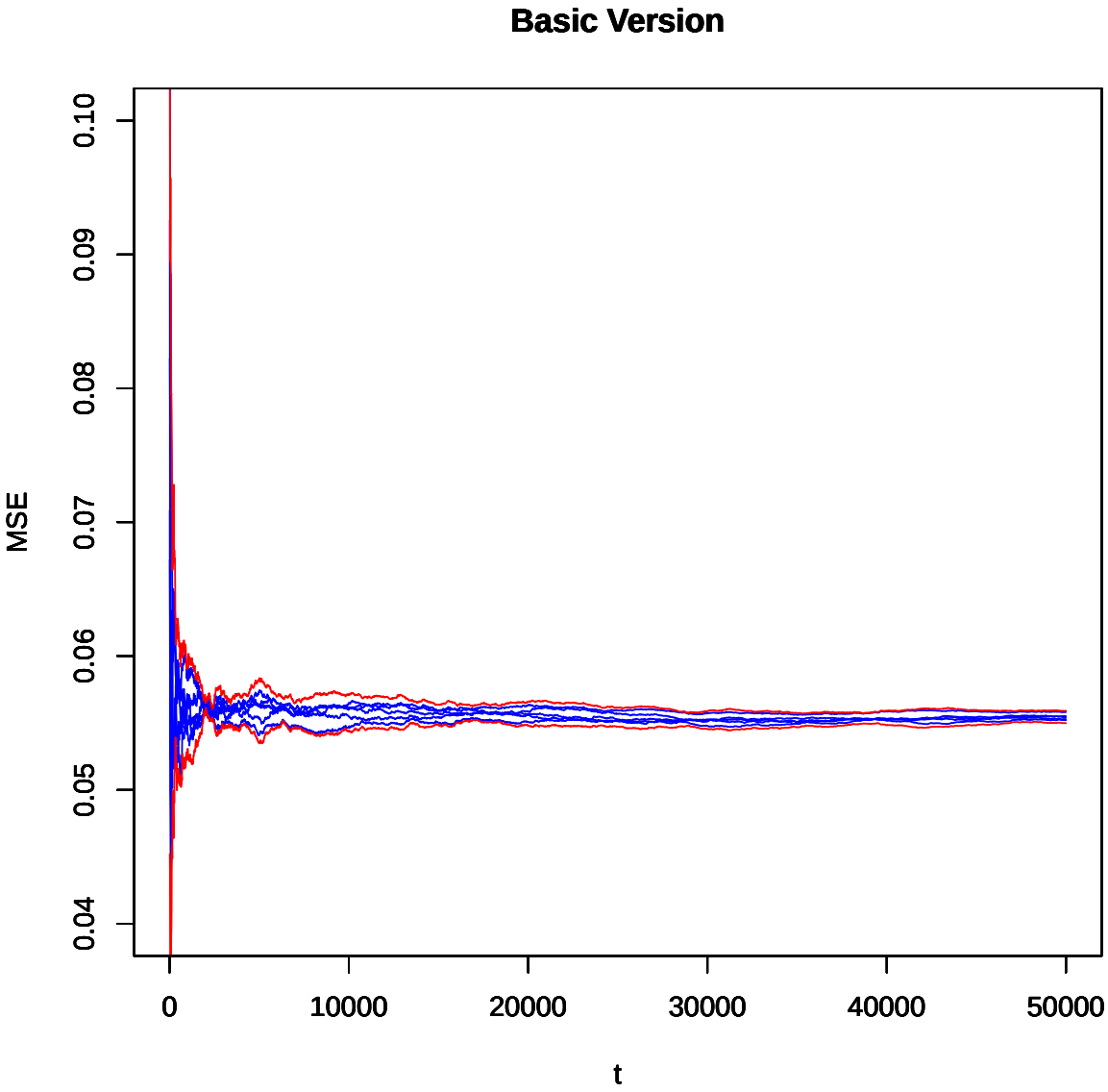

Figure 6.4: MSE when iterations rises

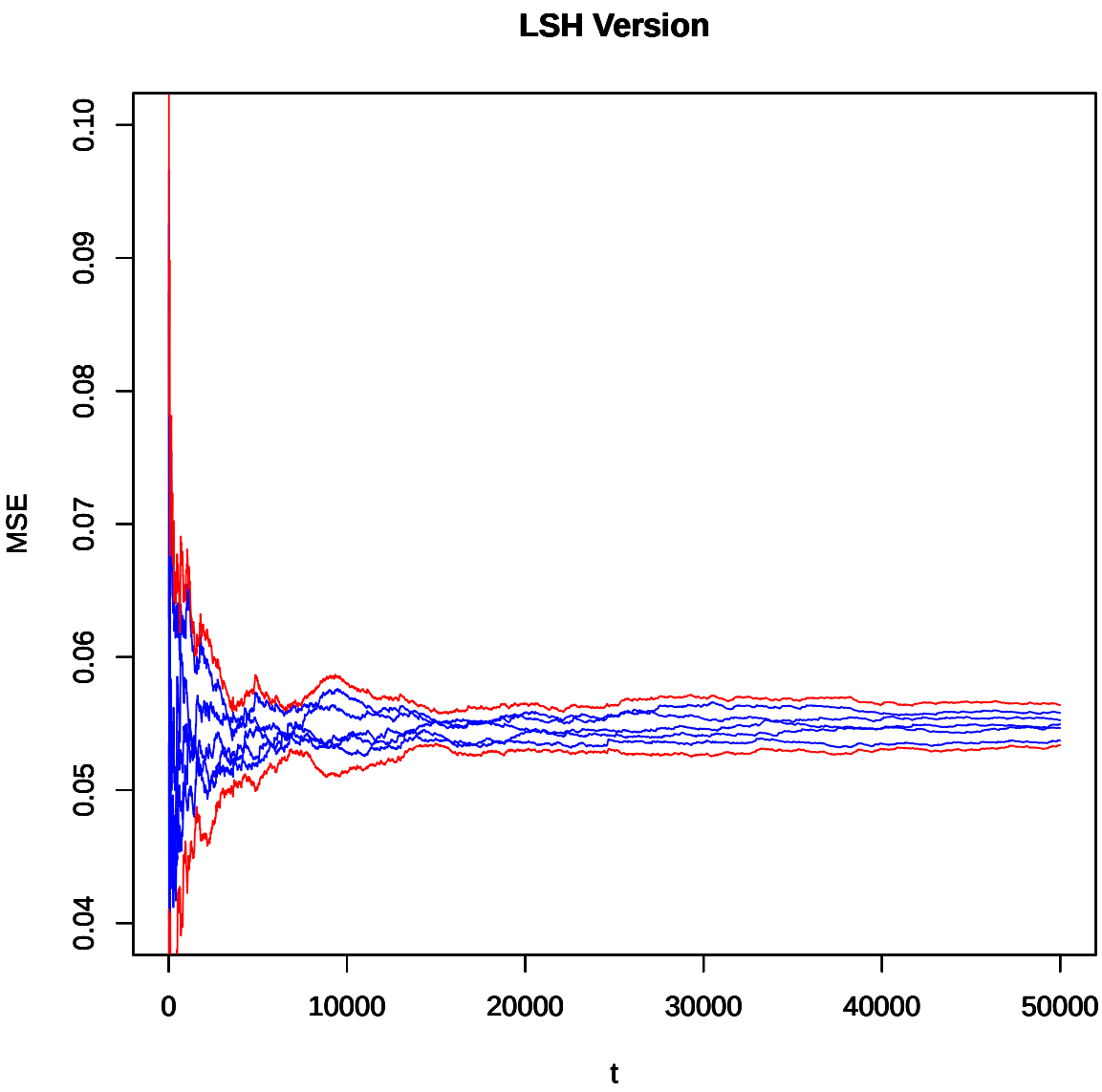

Figure 6.5: MSE when iterations rises

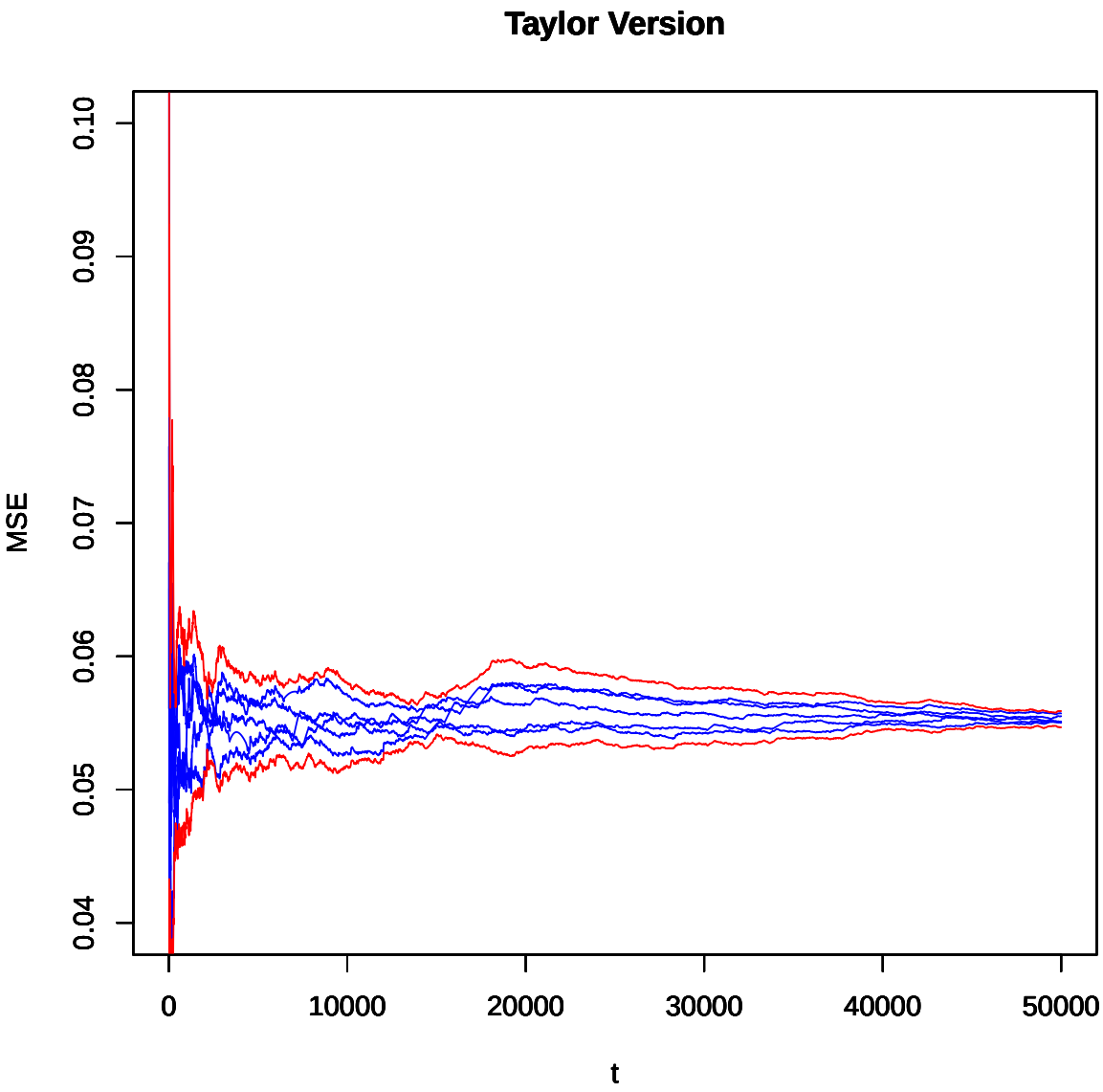

Figure 6.6: MSE when iterations rises

Next two spaghetti plots trace MSE with different k for our three algorithms. To focus on the impact of the parameter k and remove errors from Markov Chains, this time we directly draw $\theta_1, \theta_2, \ldots, \theta_n \overset{i.i.d.}{\sim} \text{Unif}(\text{Conv}(X))$ (for our simple example, it is straightforward to do this using rejection sampling). We can then use $W_k(\theta)$ as a weight for the corresponding sampled value of θ to calculate an empirical distribution $\hat{\pi}_n = \frac{1}{W}\sum_{i=1}^{n} W_k(\theta_i)\delta_{\theta_i}$, where $W = \sum_{i=1}^{n} W_k(\theta_i)$. We can then calculate mean values and mean square errors for this empirical distribution.

For the three algorithms, we also respectively observe 5 chains (blue lines) with a fixed iteration time $t = 30000$. Red boundaries present a 95% confidence interval. Both Basic version and LSH version have a tendency that MSE falls off sharply with k increasing and steadily remains in the lowest level when k continues to increase. Figure 6.9 displays linear regression functions which are selected from a bunch of regression functions. Among them, we find $MSE \propto \frac{1}{\sqrt{k}}$ has a smallest and significant p value for both versions.

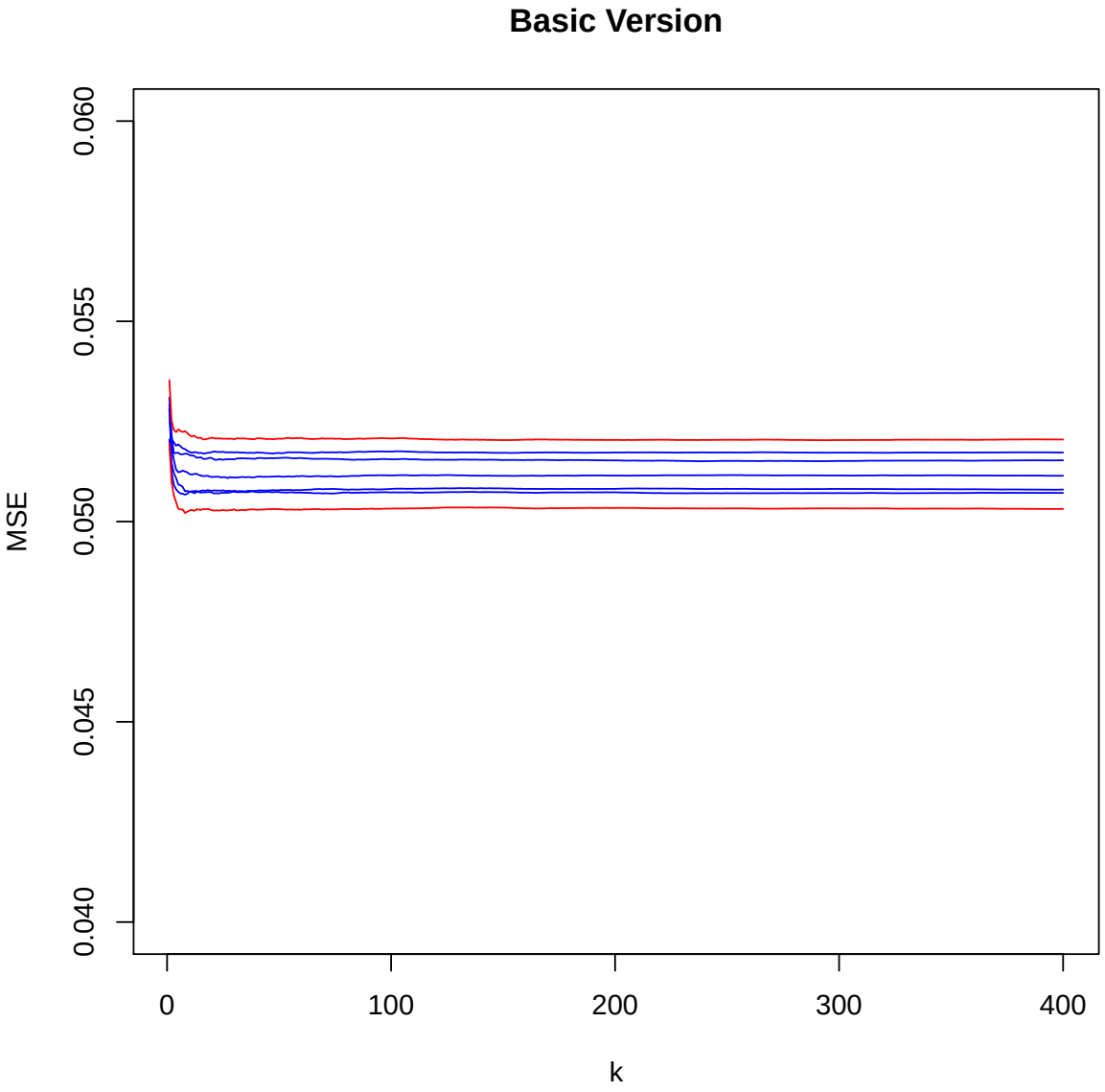

Figure 6.7: MSE with different k

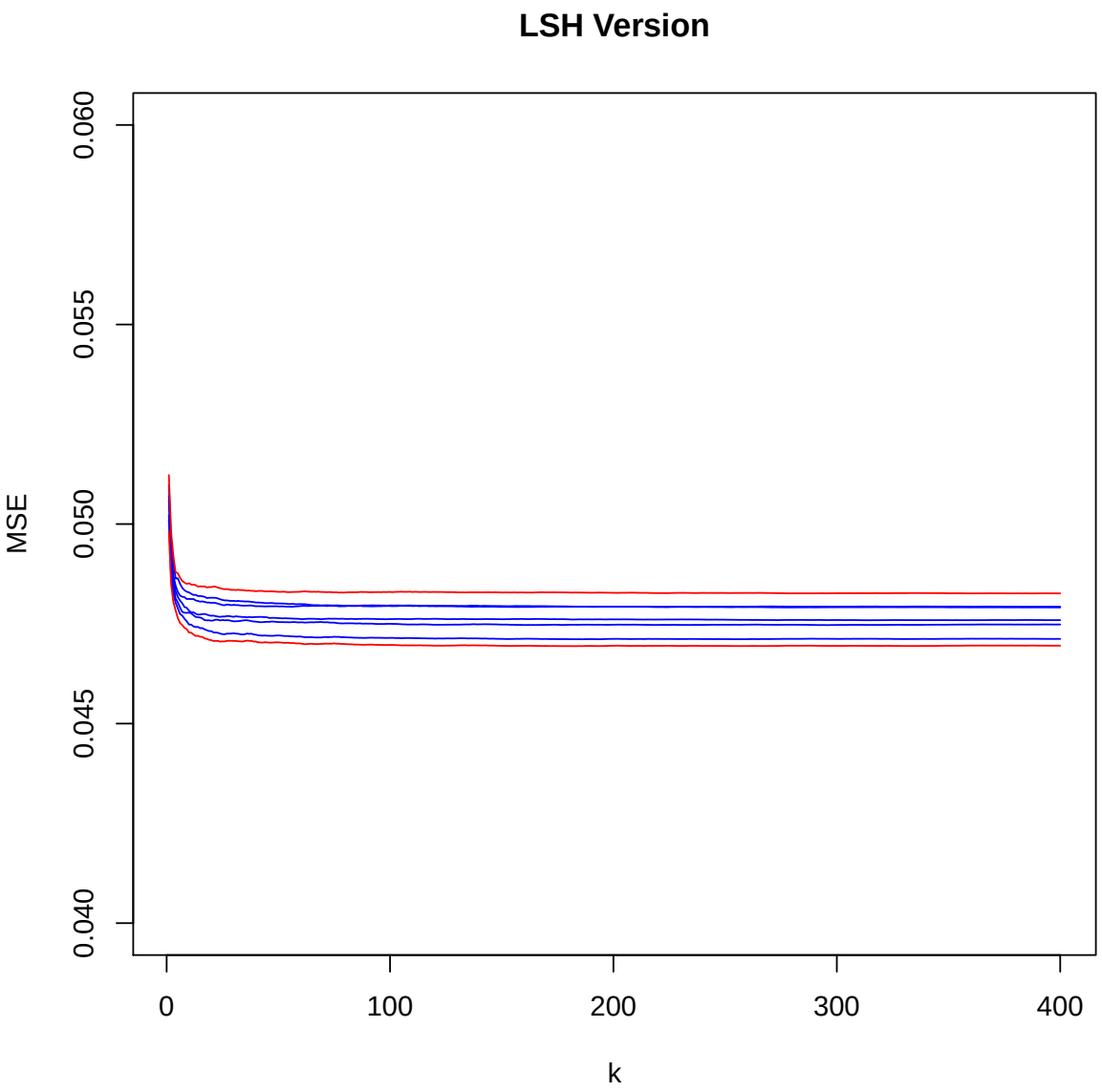

Figure 6.8: MSE with different k

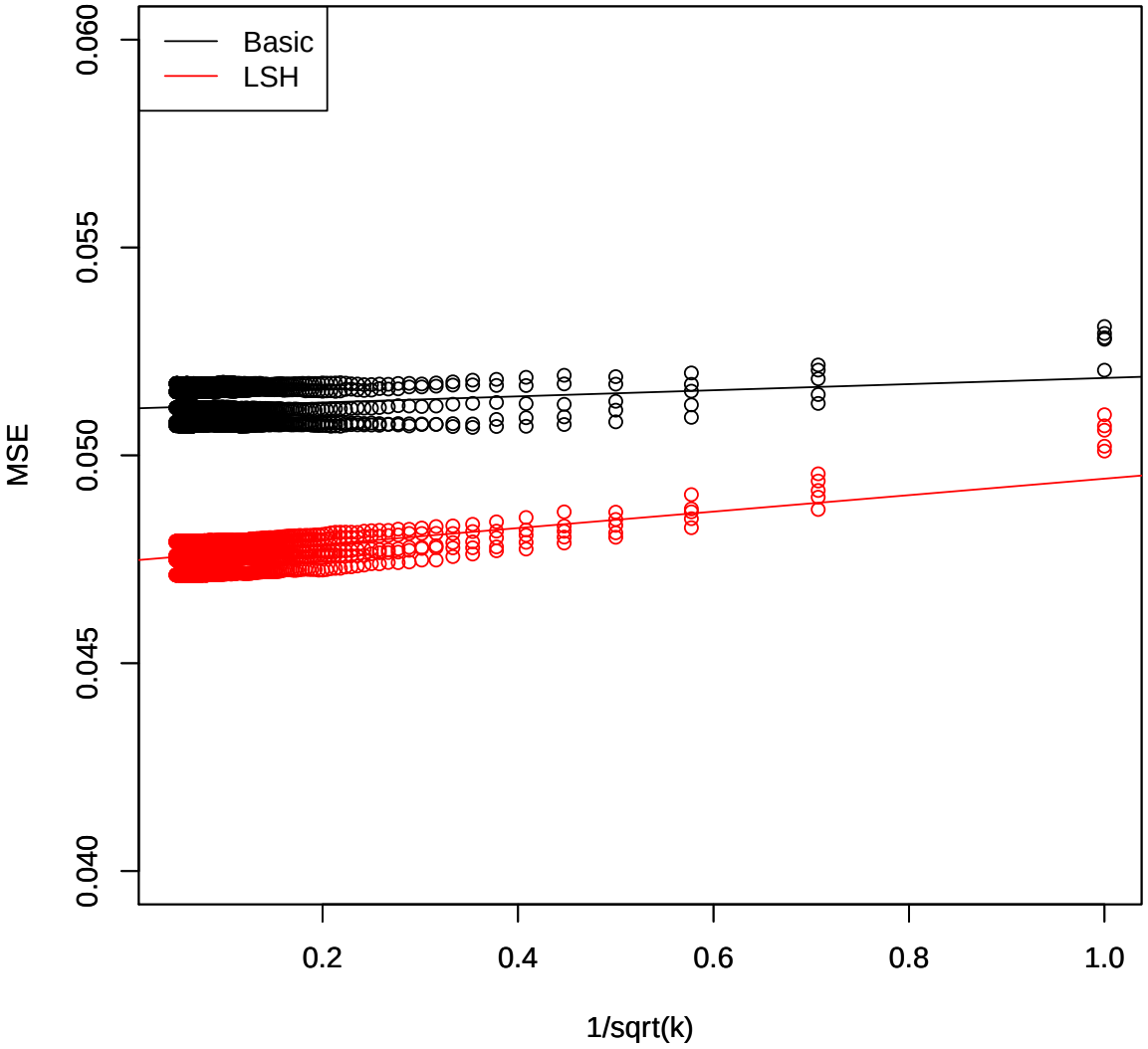

Figure 6.9: Tendency

Figure 6.10 below is a little different. MSE is essentially 0 at the very beginning and has an rising tendency later even though it quickly stop ascending and keeps steady. Actually Figure 6.11 explains the reason. Blue lines are the maximum or minimum of $W_k(\theta)$ for 5 chains. Red line in the top plot is for $\frac{1}{\sqrt{k}}$. Obviously, when k is too small like 1 or 2, $\frac{1}{k}$ is relatively far away from 0. Taylor's expansion of $W(\theta)$ actually cannot be used to approach. When minimum of $W_k(\theta)$ is bigger than $\frac{1}{k}$, some value of Taylor's expansion of $W(\theta)$ might be negative. It might rise rejection rate of Markov chains. A higher rejection

rate will also prevent transitions and remain the same samples. That is why MSE is very small and close to 0. When k is not bigger enough, Algorithm 15 can not make much sense. So, only the flat part can be used to judge if MSE converges to a low level.

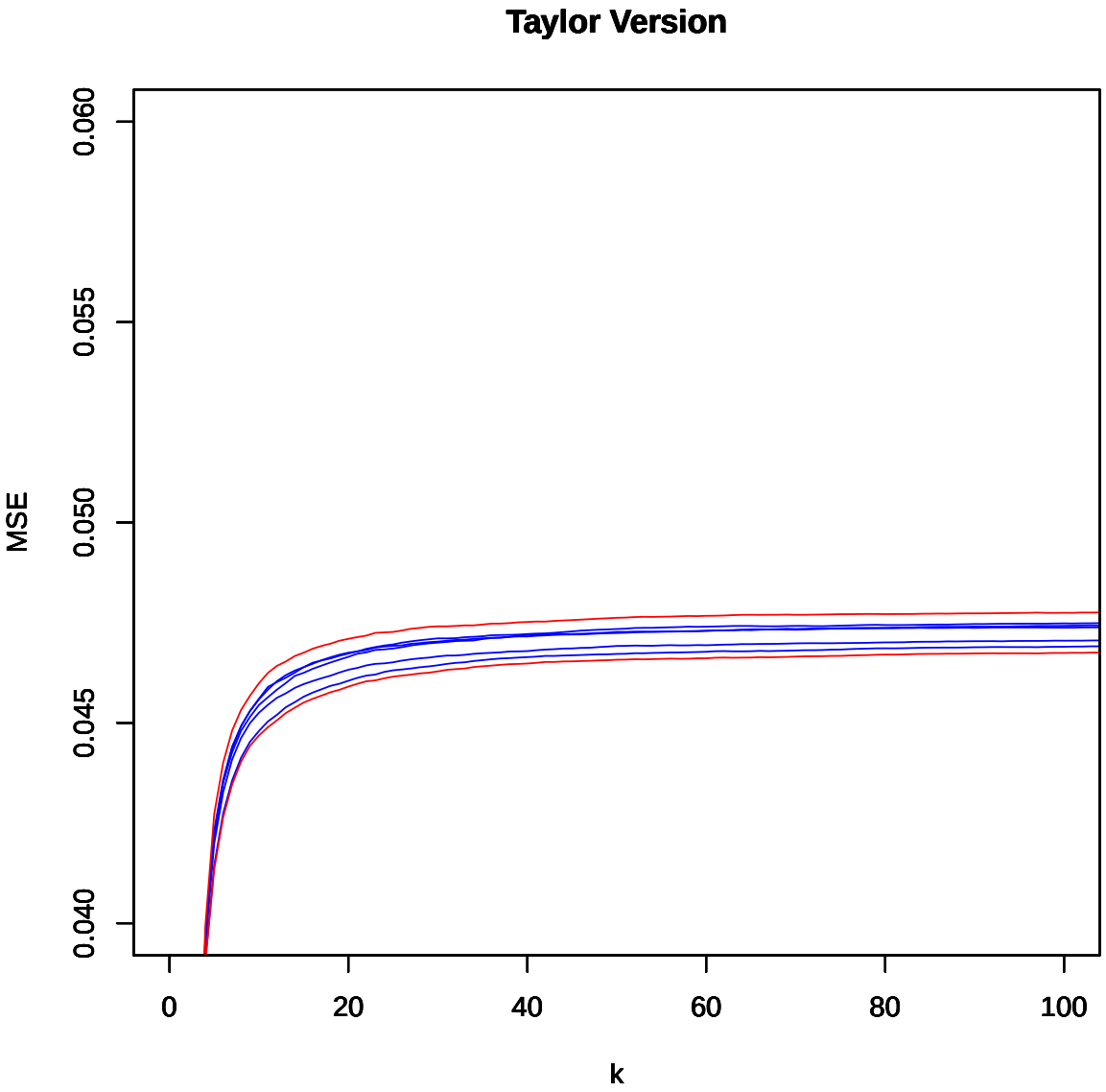

Figure 6.10: MSE with different k

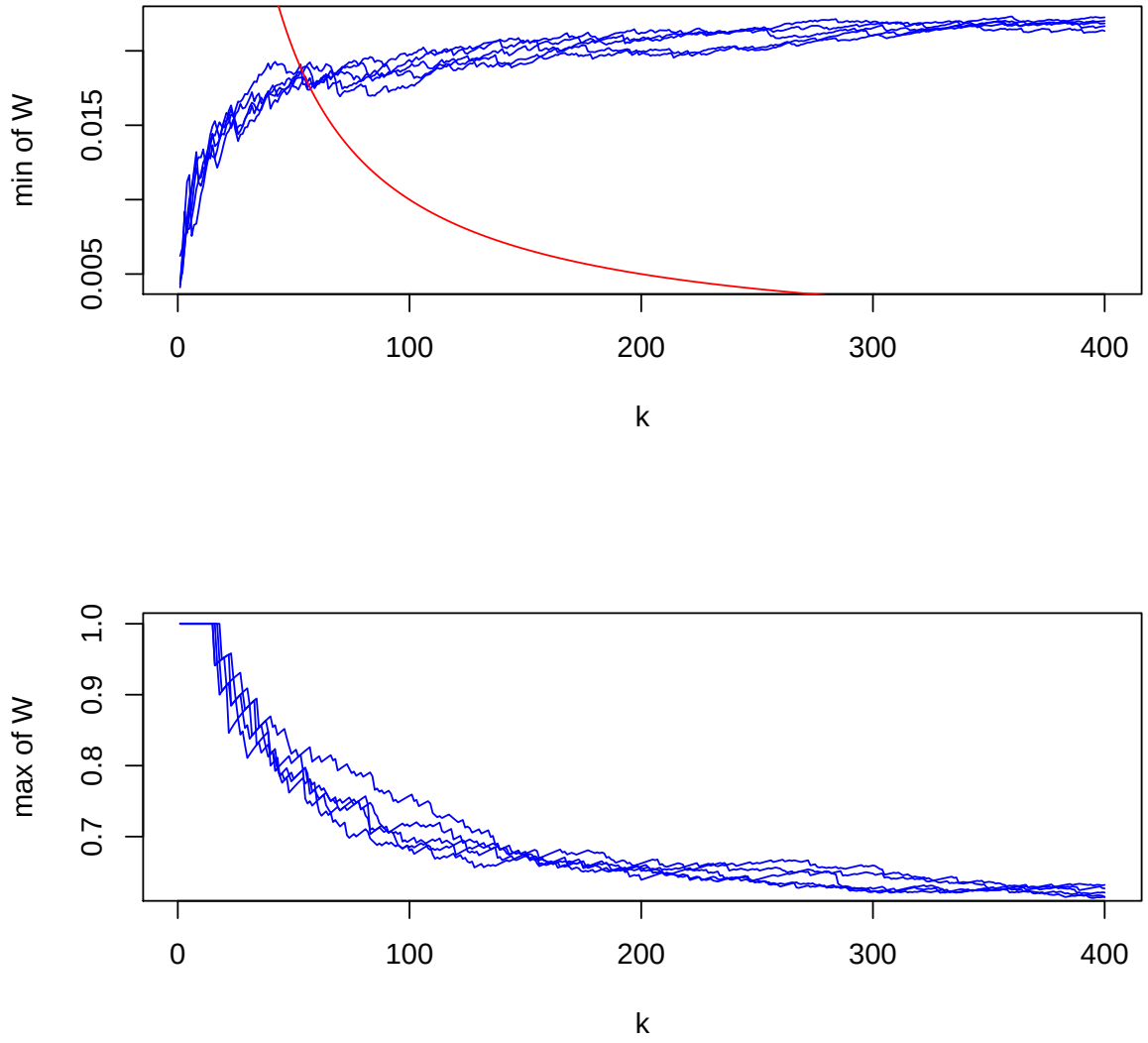

Figure 6.11: maximum and minimum of $W_k(\theta)$ with different k

6.4 Suggestions of choosing (k, n)

In this section,we would like to give some suggestions on how to match parameter k and iteration time n. Our assessment is on MSE of the difference between true and estimated densities. For every Markov chain, it can be calculated for experienced empirical distribu-

tion. We denote empirical distribution function by $\hat{f}$ and true cumulative density function (CDF) by f[1]. Then, $MSE = mean(\sum(\hat{f} - f)^2)$.

We have three hypotheses based on previous simulations and analysis. By Figure 6.9, it is suggested that $MSE \propto \frac{1}{\sqrt{k}}$. Lemma 5.1.2 also holds the same form. Errors will decrease when k increases. For iteration time n, we only know the tendency that errors will decrease when n increases. So far, we do not know what type k will match best. In practice, computers have a limited computational capability denoted by M. In our algorithms, M is roughly determined by the product of $n \times k$ because in each step of n the cost of computing $W_k(\theta)$ is linear to k according to Figure 6.2.

Take Algorithm 14 as an example. We define k and n both dependent on a variable b. Table 6.1 lists 3 probable pairs of (k, n). The product of k and n for all 3 types are equivalent. In practice, k and n should be integers. Thus, we use the integer part of $k(b)$ and $n(b)$ when they are not integers.

	type 1	type 2	type 3
k	b	$\sqrt{b}$	$log(b)$
n	$50b$	$50b^{1.5}$	$50\frac{b^2}{log(b)}$

Table 6.1: scale relationship

Figure 6.12 shows the MSE of the difference between true marginal density and our estimator. Different combinations of (n, k) are listed in Table 6.1. Type 1 and Type 2 perform better and converge to 0. Especially, Type 2 has lower variance and converges to 0 faster. So we think Type 2 has the best performance.

[1] f is not a real CDF. There is a sharply down line after 0.996. In this interval, the expression of our density function is quite hard to integrate. So we set the value of θ in this interval equals to 1. They should be bigger than 0.998. We think it is acceptable to ignore the tiny bias.

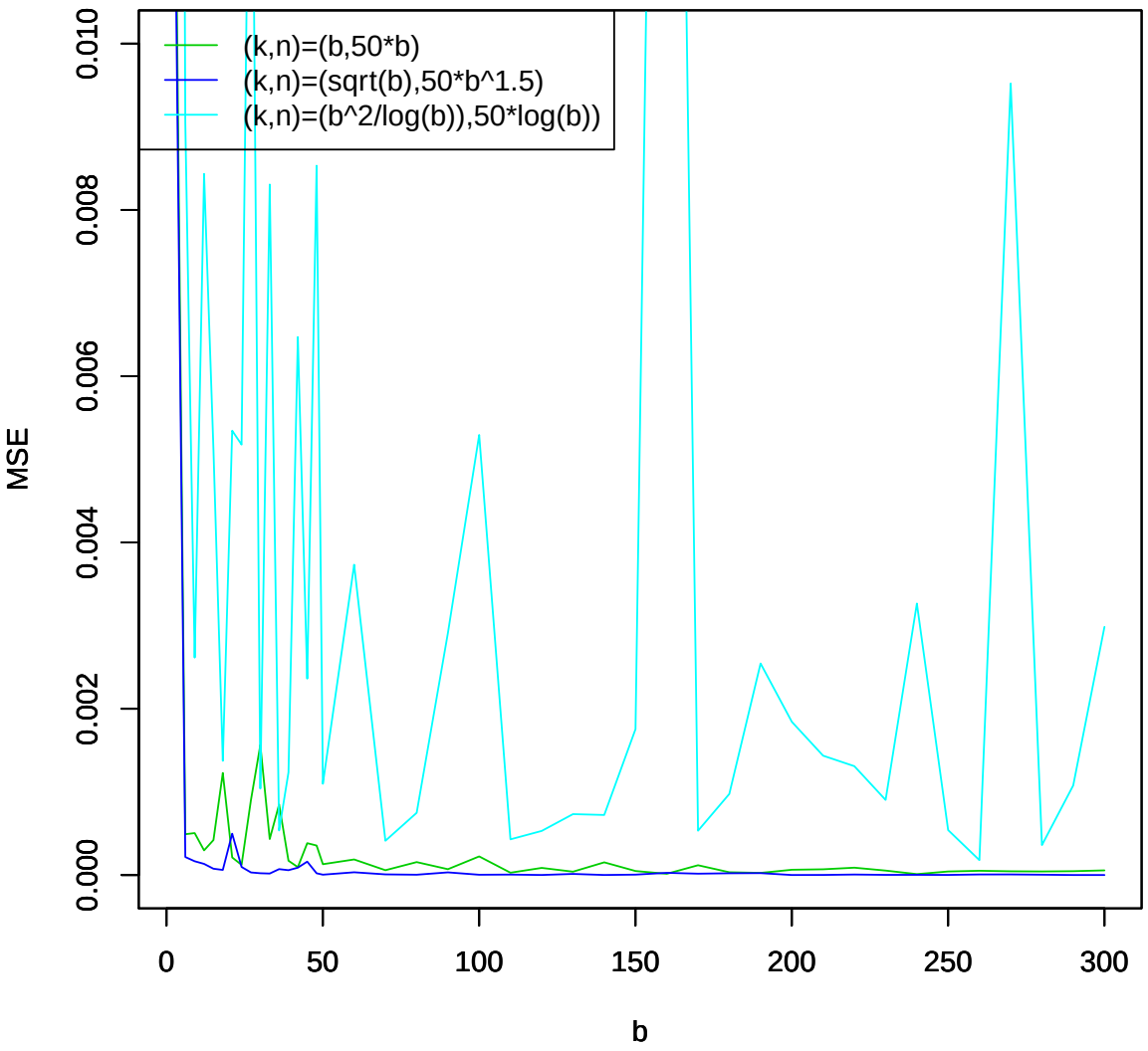

Figure 6.12: Model Error with respect to relationship

When we hold this good scale relationship between k and n, we would like to continually test 2 different coefficient relationships listed in Table 6.2.

	Co. 1	Co. 2
k	$\sqrt{b}$	$5\sqrt{b}$
n	$50b^{1.5}$	$10b^{1.5}$

Table 6.2: Various Coefficients

Figure 6.13 shows the MSE of the difference between true marginal density and the estimated. We find when the coefficient of n is as 50 times as k, our algorithm mixed more quickly and has a lower MSE at the same time.

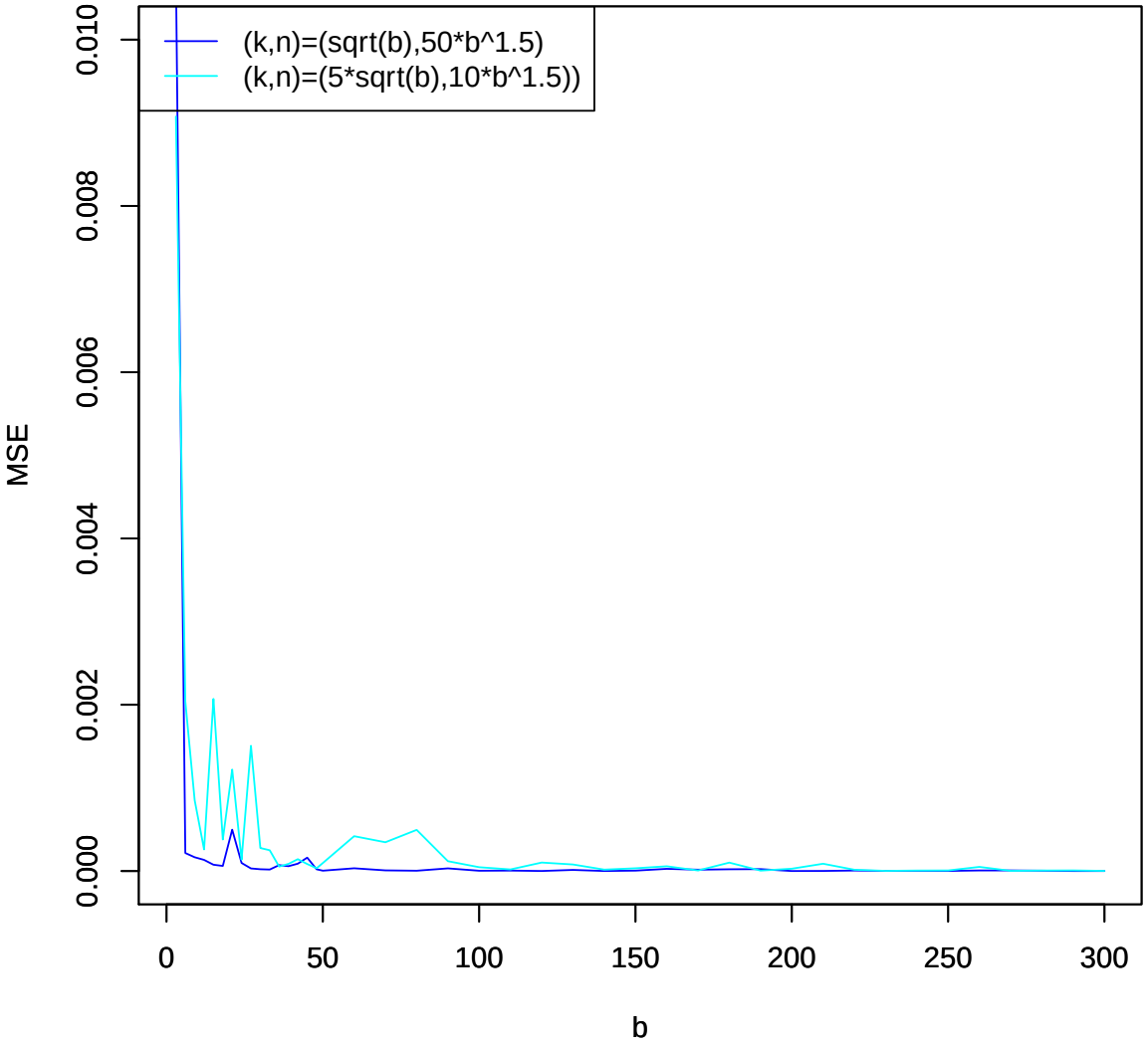

Figure 6.13: Model Error with respect to coefficients

In a conclusion, when $(k, n) = (\sqrt{b}, 50b^{1.5})$, our algorithms will get best efficiency. Meanwhile, it has a lower level fluctuation.

Chapter 7

Conclusion

The first six chapters completely cover the background, solution and simulation of the problem we focus on. Our algorithms can work for approaching to a uniform distribution on convex hulls. And it is feasible to put our algorithms into practice because we do not limit the structure of convex hulls. However, the method of decomposing the convex hulls and separating off small sub convex hulls actually depends on cases. It is beyond the scope of this thesis.

Definitely, our algorithm has some shortages. The biggest one is LSH and Taylor's expansion do not perform as well as we expect. Maybe it is affected by pseudo marginal algorithm's sensibility or we actually use approximation on both of them. But we fail to find the exact reason. In our experiments, although time consumption actually reduce, there are some bad results with bigger variance than we present. It is quite worthwhile to find out deeper reasons. If time permit, we can also try to adjust LSH and Taylor's expansion to fit our algorithm better.

References

[1] Christophe Andrieu and Gareth O Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

[2] Imre Bárány and Zoltán Füredi. Computing the volume is difficult. *Discrete & Computational Geometry*, 2(4):319–326, 1987.

[3] Daniel G Brown. How I wasted too long finding a concentration inequality for sums of geometric variables. *Found at https://cs. uwaterloo. ca/˜ browndg/negbin. pdf*, 6, 2011.

[4] CJ Burke and M Rosenblatt. A Markovian function of a Markov chain. *The Annals of Mathematical Statistics*, 29(4):1112–1122, 1958.

[5] George Casella, Christian P Robert, Martin T Wells, et al. Generalized accept-reject sampling schemes. In *A Festschrift for Herman Rubin*, pages 342–347. Institute of Mathematical Statistics, 2004.

[6] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

[7] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[8] Charles J Geyer. Practical Markov Chain Monte Carlo. *Statistical science*, pages 473–483, 1992.

[9] Pierre E Jacob, Alexandre H Thiery, et al. On nonnegative unbiased estimators. *The Annals of Statistics*, 43(2):769–784, 2015.

[10] Liu Jianping. MCMC(1) Monte Carlo Method. https://www.cnblogs.com/pinard/p/6625739.html. Accessed April 27, 2017.

[11] Anoop Korattikara, Yutian Chen, and Max Welling. Austerity in MCMC Land: Cutting the Metropolis-Hastings Budget. *Preprint*, 2013.

[12] Dougal Maclaurin and Ryan P Adams. Firefly Monte Carlo: Exact MCMC with subsets of data. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[13] Nicholas Metropolis and Stanislaw Ulam. The Monte Carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[14] Matias Quiroz, Mattias Villani, and Robert Kohn. Scalable MCMC for large data problems using data subsampling and the difference estimator. 2015.

[15] Murray Rosenblatt. A central limit theorem and a strong mixing condition. *Proceedings of the National Academy of Sciences of the United States of America*, 42(1):43, 1956.

[16] Ming-Hui Chen Bruce Schmeiser. Performance of the Gibbs, Hit-and-Run, and Metropolis Samplers. 1992.

[17] Miklós Simonovits. How to compute the volume in high dimension? *Mathematical programming*, 97(1-2):337–374, 2003.

[18] Robert L Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.

[19] Martin A Tanner and Wing Hung Wong. The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association*, 82(398):528–540, 1987.

[20] Luke Tierney. Markov chains for exploring posterior distributions. *the Annals of Statistics*, pages 1701–1728, 1994.

[21] M-N Tran, R Kohn, M Quiroz, and M Villani. Block-wise pseudo-marginal Metropolis-Hastings. 2016.

[22] Theodore A Vessey. A Modern Approach to Probability Theory. *The American Mathematical Monthly*, 104(8):787, 1997.