Efficient Monte Carlo Random Sample Generation Through Discretization

by

Chel Hee Lee

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

Master of Science

Department of Statistics

University of Manitoba

Winnipeg, Manitoba, Canada

Efficient Monte Carlo Random Sample Generation Through Discretization

By

Chel Hee Lee

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

Master of Science

Contents

List of Figures

List of Tables

Acknowledgements

Dr. Liqun Wang has been a source of encouragement to me while working in the Master's program at the University of Manitoba. I have had the opportunity and great pleasure of interacting with Dr. Wang; his door has always been open to me. From Dr. Wang's insight and experience, I have developed a keen appreciation for computational statistics.

I would also like to express my heartfelt appreciation to Dr. Ruppa Thulasiram in the department of Computer Science for sharing his ideas and interests for this sampling algorithm from the computational perspective.

Dr. James C. Fu, Dr. John Brewster and Dr. Alexandre Leblanc have provided statistical advice and have assisted me in identifying any areas of misunderstanding and provided the resources to pursue my interests.

A simple thank you does not cover the debt of gratitude I owe to Dr. Lisa Lix. Her constant encouragement has kept me focused on the successful completion of this thesis.

I am also grateful to Drs. Lisa Lix and Liqun Wang for the financial support through their joint research grant from Canadian Institute of Health Research (CHIR); to Dr. Liqun Wang for support through his grant from Natural Sciences and Engineering Research Council of Canada (NSERC); and lastly to the University of Manitoba Faculty of Science for the Science Scholarship.

Abstract

This thesis develops an efficient discretization based Monte Carlo algorithm studied by Wang and Lee (2008) for generating random variates from high dimensional distributions of complex structures. The improvement to a discretization-based algorithm of Fu and Wang (2002) is achieved regarding the computational efficiency. The cornerstone of this algorithm is the discretization of the sample space and efficient numerical inversion of a multivariate cumulative distribution function. This algorithm is dimension-free, non-iterative, and easy to implement. These characteristics compensate for some limitations of Markov chain Monte Carlo (MCMC) methods and will be featured using classic examples and applications. For general use, a practical guide with programs implemented by GNU statistical software R is provided for practitioners.

Chapter 1

Introduction

The studies of Fu and Wang (2002) and Wang and Fu (2007) provided a practical computational algorithm for multivariate random sample generation. This algorithm overcomes many limitations of the Markov chain Monte Carlo (MCMC) methods and successfully identifies the quantity of interest in a statistical inference with model complexity in high dimensions. In this thesis, it will be referred to as the Fu-Wang algorithm and its principles will be investigated in Chapter 3. An improved algorithm, henceforth in this thesis entitled the Wang-Lee algorithm, will be developed in the following area: data structure, sorting, searching, and different approaches in numerical analysis based on this examination of the Fu-Wang algorithm. This is a methodological work which uses a sample-based simulation study to efficiently generate multivariate random samples from high dimensional distributions with elaborate structure.

1.1 Domain of Problems

A computational challenge is predominantly a distinguished issue in statistical problem-solving. This statistical problem-solving typically involves a procedure from statistical modelling on real data to a statistical inference for the best statistical

model selection and its prediction. The best model is determined on the basis of how rationally and precisely a designed model explains and describes the real data. It is essential to efficiently esimate parameters in statistical problem-solving. In this section, the etiology of the computational burden will be discussed on the statistical inference. A similar argument in Section 1.1 is also put forth by Robert and Casella (2004)

1.1.1 Statistical Inference in Complex Models

A well designed statistical model usually includes many independent variables to clarify the real data. However, the mathematical form in the designed model will become more complex as the number of independent variables increases. The model complexity has a direct connection with the type of model and its dimensions. In the statistical inference, these are often considered as the primary sources resulting in the failure of the analytical estimation of interest. Another contributing cause to the failure of the analytical evaluation of interest is computer related limitations (see Section 4.1).

Examples of this type of problem are found from a Gaussian mixture model and a Bayesian hierachical model. With respect to the computation of a mixture model specifically, Robert and Casella (2004) clearly state that "the representation of the likelihood function (and therefore the analytical computation of maximum likelihood or Bayes estimates) is generally impossible for mixtures" (p. 4). Therefore the analytical computation of maximum likelihood or Bayes estimates is considered implausible.

However, Wang and Fu's 2007 study provided successful results for the statistical quantity of interest with this model; the statistical model in their study utilized a mixed form of Gaussian mixture and Bayesian hierarchical models, as shown in the

equation (1.1).

$$f(y) = \prod_{j=1}^{K} w_j f_j(y|\mu_j, \sigma_j^2), \tag{1.1}$$

where y is observations from $y \sim f(y|\mu, \sigma^2)$, K is the number of sub-populations, $w_j > 0$ is the probabilites of sub-populations and $\sum w_j = 1$. (Wang & Fu, 2007, p. 633). This equation clearly illustrates the model complexity regarding the number of parameters to be estimated on the statistical modelling as the number of sub-population increases.

1.1.2 Numerical Optimization and Integration

With regard to the statistical inference in the mixture model mentioned in Section 1.1.1, $\mathcal{O}(kn)$ and $\mathcal{O}(k^n)$ computations are needed on the inference by a classical likelihood and a Bayesian method respectively (Robert & Casella, 2004, p. 4). The connection exists that the computational burden will be dramatically increased as either the number of k(sub-population) or n(sample size) increases. These represent two conventional streams of statistical methodologies in the statistical inference.

Maximum likelihood estimator (MLE) is a value which maximizes the likelihood function of $f(x)$, $L(\theta|x)$, or the log-likelihood function, $\ln L(\theta|x)$, when assuming a set of observations, $(x_1, x_2, \ldots, x_n)$ and a general density, $f(x)$, are given. In consideration of the computational convenience, its solution, $\hat{\theta}$, is easily found by differentiating $\ln L(\theta|x)$ about θ, where $\theta \in \Omega$ and Ω is a parameter space. Hence, it can be preceived as the opitimization problem. The study of Wang and Fu (2007) cannot be solved analytically using conventional numerical methods due to its model complexity and numerous computations. Its likelihood function is identified as the following equation (1.2) (Wang & Fu, 2007, p. 634).

$$\prod_{i=1}^{N} f(y_i|\mu^K, \sigma_K^2, w^K, K) = \prod_{i=1}^{N} \sum_{j=1}^{K} \frac{w_{Kj}}{\sqrt{2\pi\sigma_K^2}} \exp - \frac{(y_i - \mu_{Kj})^2}{2\pi\sigma_K^2} \tag{1.2}$$

The computational challenge is also revealed from the definition of posterior function on a Bayesian paradigm. Suppose that $\pi(\theta)$ is prior information and a set of observations, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, and $f(\mathbf{x}|\theta)$ is a sample distribution. The posterior density function, $p(\theta|\mathbf{x})$, then, is:

$$p(\theta|\mathbf{x}) = \frac{f(\mathbf{x}|\theta)\pi(\theta)}{\int f(\mathbf{x}|\theta)\pi(\theta)} = \frac{f(\mathbf{x}|\theta)\pi(\theta)}{m(x)} \tag{1.3}$$

where $m(x) = \int f(\mathbf{x}|\theta)\pi(\theta)d\theta$ is a marginal distribution of $\mathbf{x}$. In the denominator of equation (1.3), the form of integration exists as the marginal distribution, $m(\mathbf{x})$. A significant issue is how to solve this integration in high dimensions with as complex structure as the equation (1.4) in a study of Wang and Fu (2007) with prior specification of μ, σ^2, w, and K (p. 635).

$$\prod_{i=1}^{N} f(y_i|\mu^K, \sigma^2, w^K, K)p(\mu^K|K)p(\sigma^2|K)p(w^K|K)p(K) \tag{1.4}$$

According to Robert (2007), the increase of number of dimensions leads to the rapid loss of numerical accuracy regardless of which typical numerical method is used. Robert stresses that "most standard methods should not be used for integration in dimensions larger than 4" (p. 293). It is the strong argument for further investigation of the Fu-Wang algorithm with this type of model complexity with high dimensions.

1.2 Previous Research

In Section 1.1.1 and Section 1.1.2, significant sources of computational difficulties in statistical inference are established; the integral of non-closed from a likelihood or posterior distribution. In particular, Robert and Casella (2004) note it as "a practical realization of the curse of dimensionality" (p. 268).

Concerning this kind of impractical computation, "Markov chain Monte Carlo

(MCMC) is a key technique for calculating analytically intractable integrals in high dimensions" (Sahu & Zhigljavsky, 2003, p. 395).

Numerous statisticians have identified limitations on MCMC methods and are presently employed in their resolution. As reported by Fishman (2006), the correlated samples generated by MCMC are sequential as the intrinsic property of MCMC (p. 203). However, the matter of highly correlated samples in MCMC generate another restriction — "a slow mixing Markov chain" (W. R. Gilks, Roberts, & Sahu, 1998, p. 1045). The number of iterations has an inverse association between the cost and numerical accuracy (Fishman, 2006, p. 203; W. R. Gilks et al., 1998, p. 1045; Liu, Liang, & Wong, 2001, p. 561). Reparameterization is essential on occasion for the successful implementation of MCMC (Fu & Wang, 2002, p. 6, W. Gilks, Richardson, & Spiegelhalter, 1998, p. 97). A starting point is the influential factor on the rate of convergence (Robert, 2007, p. 302). The limitation of the standard MCMC in consideration of examining posterior modes is attributed to the localization tendency of MCMC algorithm (Celeux, Hurn, & Robert, 2000, p. 957).

1.3 Thesis Organization

This thesis is structured in the following way. Chapter 2 briefly explains some preliminaries essential to understanding the Wang-Lee algorithm. Chapter 3 analyzes the procedures of the Fu-Wang algorithm. Also, limitations of the Fu-Wang algorithm which lead to the development of the Wang-Lee algorithm are discussed. Chapter 4 summarizes a computational algorithm at each stage of the Wang-Lee algorithm. Chapter 5 demonstrates the Wang-Lee algorithm using examples. A guide for use of the Wang-Lee algorithm is provided. Chapter 6 shows the validity of the Wang-Lee algorithm using comparisons to standard applications. Finally, Chapter 7 concludes with a discussion of concerns associated with the use of the Wang-Lee algorithm and

suggests resolutions to the constraints of this algorithm. Chapter 7 also proposes areas for further research to enhance fundamental applications.

Chapter 2

Random Variates Generation

Generally speaking, an algorithm can be regarded as a sequential combination of several statistical procedures. Essential statistical preliminaries will be covered by Chapter 2 to provide a basis for understanding of the Fu-Wang algorithm in Chapter 3 as well as the Wang-Lee algorithm in Chapter 4.

The objective of these algorithms is to efficiently generate random variates from a given or target density function, $f(x)$, from a significant region. It also implies that crucial techniques in this algorithm are mainly categorized into a random variate generation and a variance reduction.

First, a given or target density function, $f(x)$, may be one of a simple standard distribution and/or a complex non-standard distribution. Typically most commercial statistical software provides libraries with which to generate random variates from the standard distributions. However, it is not easy to obtain non-standard random variates from non-standard distributions from the libraries provided by this software. To accomplish this, it is necessary to understand how to generate random variates from a given or target density function, $f(x)$, not subject to the standard distribution through Section 2.1.

Secondly, the basic idea of significant region is to draw more samples from a high

density area rather than a low density area. According to the following criteria from Gentle, 2002, p. 59: "an objective is to devise a sampling plan that will yield estimators with small variance", it proves the fact that the Fu-Wang algorithm is well constructed as a sampling algorithm. For the purpopse of variance reduction, the importance sampling and the stratified sampling techniques are reviewed in Section 2.2.

In the computational perspective random variates generation and its simulation study, three major facts should be known in advance as the random variates are derived from the psuedo random numbers. First, a sequence of random numbers should be reproducible under the equivalent conditions of a Monte Carlo study (Gentle, 2003, p. 230). Secondly, investigation of a sampling algorithm is necessary since "total computing time can be substantial and sample generation accounts for more than 80% of this time" (Fishman, 2006, p. 74). Finally, the quality of uniform distribution should not be overlooked because any random numbers generated from the computer are on the basis of a behaviour of uniform distribution (Fishman, 2006, p. 75; Robert & Casella, 2004, p. 39).

Several excellent references related to Chapter 2 include the book *Non-uniform Random Variate Generation*, written by Devroye (1986). This book includes useful algorithms for generating random deviations from non-standard distributions. Numerical Recipes, by Press, Teukolsky, Vetterling, and Flannery (2007) is a masterpiece in the field of numerical analysis. This practical text provides the complete codes in C/C++, efficient program techniques, and detailed mathematical solutions for statistical algorithm development (See Chapter 7.)

2.1 Generating Random Variates

There are two methodologies commonly used: 1. Inverse-CDF Method, and 2. Acceptance-Rejection Method.

2.1.1 Inverse CDF Method

Any random variates from a given or target density function, $f(x)$, are obtained by an inverse function of its cumulative density function, $F^{-1}(U)$, if and only if, U is a standard uniform distribution. The following Lemma is the mathematical formulation of the inverse CDF Method (Robert & Casella, 2004, p. 39).

Lemma 2.1.1. *If $U \sim U[0,1]$, then the random variable $F^{-1}(U)$ has the distribution F.*

For example, if $F(x)$ is a non-decreasing and one-to-one function, then

$$\begin{aligned} F_U(u) &= P(F(X) \leq u) \\ &= P(X \leq F^{-1}(u)) \\ &= F(F^{-1}(u)) \\ &= u. \end{aligned}$$

The general procedure of the inverse transformation in the case of discrete random variable X is described in the Algorithm 2.1.1 (Ross, 1996, p. 45)

Consider a discrete random variable $X \sim P(X = x_j)$ where $\sum_j^{\infty} p_j = 1$ and $P(X = x_j) = P(\sum_{i=1}^{j-1} \leq U < \sum_{i=1}^{j}) = p_j$

Algorithm 2.1.1 Inverse CDF Method - Discrete Distribution

$$X = \begin{cases} x_0 & \text{if } U \leq p_0 \\ x_1 & \text{if } p_0 \leq U \leq p_0 + p_1 \\ \vdots & \\ x_j & \text{if } \sum_{i=1}^{j-1} p_i \leq U \leq \sum_{i=0}^{j} p_i \\ \vdots & \end{cases}$$

2.1.2 Acceptance-Rejection Method

For the purpose of multivariate random variates generation, the Acceptance-Rejection method is widely used. The general procedure of the Acceptance-Rejection method is described in the Algorithm 2.1.2 (Ross, 1996, p. 54)

Consider a random variable $X \sim p_j = P(X = j)$ where $j = 1, 2, \ldots, \infty$.

Algorithm 2.1.2 Acceptance-Rejection Method

Step 1. Genereate $Y \sim q_j$

Step 2. Generate $U \sim U[0, 1]$

if $U < p_Y/cq_Y$ **then**

Set $X = Y$

else

Return to Step 1.

end if

Here, the choice of constant is $c = \sup f(x)/g(x)$ (Madras, 2002, p. 21).

2.2 Variance Reduction Techniques

This variance reduction technique is essential for the use of the Fu-Wang algorithm and the Wang-Lee algorithm. In statistical inference, it is of primary importance to locate an estimate with a small bias and its minimum variance. This is the foundation for locating the significant region in the Fu-Wang algorithm. It implies that samples are drawn from a concentrated area of density function. Drawn samples from the significant region provide smaller variance rather than from the independent random sampling.

A direct connection exists between samples drawn from the significant region and the graphical dignostic tool (see Section 5.3 for a detailed explanation using the Wang-Lee algorithm).

Two techniques typically used for both variance reduction and sampling efficiency are Importance Sampling and Stratified Sampling.

2.2.1 Importance Sampling

Importance sampling is "a weighted sampling" (Robert & Casella, 2004, p. 90) to draw samples from a more important region; samples should be drawn from the important region as opposed to the negligible region for the optimal sampling. The following equations (2.1) and (2.2) are the definition of Importance Sampling and its optimal importance function (Gentle, 2002, pp.59–60).

Suppose that a function, $f(x)$, on the domain $\mathcal{D}$ can be decomposed into any function, $g(x)$, and its probability function, $p(x)$, subject to $\sum p(x) = 1$. Then,

$$\theta = \int_{\mathcal{D}} f(x)dx = \int_{\mathcal{D}} \frac{f(x)}{p(x)} p(x)dx \tag{2.1}$$

where $p(x)$ is the importance function on the domain $\mathcal{D}$. Its optimal choice of importance function is

$$p(x) = \frac{|f(x)|}{\int_{\mathcal{D}} |f(x)|dx} \tag{2.2}$$

2.2.2 Stratified Sampling

Staratified Sampling is simply one of the techniques used to reduce the variance by dividing the population into several homogenous sub-populations. By this sampling technique, several good results are obtained as follows: 1. good representation of the population, 2. comparison of sub-populations, 3. cost efficiency, and 4. smaller variance rather than that obtained using simple random sampling (Lohr, 1999, pp. 95–96).

The following theorem provides the evidence of variance reduction by use of a stratified sampling (Madras, 2002, p. 33).

Theorem 2.2.1. *If $n_i = na_i$ for $i = 1, \ldots, M$, then the stratified estimator has smaller variance than the simple sampling estimator $\hat{I}_n$. In fact,*

$$var(\hat{I}) = var(T) + \frac{1}{n}\sum_{i=1}^{M} a_i \left(\frac{I_i}{a_i} - I\right)^2$$

where $a_i = P(X \in \mathcal{S}_i)$ subject to $\sum_{i=1}^{M} a_i = 1$ and $\mathcal{S} = \cup_{i=1}^{M} \mathcal{S}_i$.

Chapter 3

Fu-Wang Algorithm

In a multivariate random sample generation from a non-standard distribution in high dimensions, MCMC methods are determined to be a good solution despite inherent limitations (see Section 1.2). Fu and Wang (2002) initiated development of a new practical algorithm in order to overcome major difficulties of MCMC. In particular, difficulties "associated with multi-modality of the underlying distribution, ill-shaped sample space, as well as convergence of the iterative process." (p. 6). Wang and Fu (2007) showed its practical utility on the mixture model with unknown components. As the Wang-Lee algorithm originates from this Fu-Wang algorithm, it is necessary to investigate their studies carefully before demonstrating the Wang-Lee algorithm. In Section 3.1, the Fu-Wang algorithm is clearly identified on each step of sampling procedure.

(For ease of comparison and enhanced understanding of the Wang-Fu algorithm and the Wang-Lee algorithm, all notations and terminologies in Chapter 3 are entirely identical to those used in a study of Wang and Fu (2007).)

3.1 Sampling Procedure

3.1.1 Initial Compact Cover

Suppose that a general density function, $f(x)$, up to the multiplicative constant in d-dimensions is given. The Fu-Wang algorithm begins with setting bounds of support. It is assumed that bounds of support include the signficant region. Let $C_0(f)$ be the initial compact set subject to $C_0 \subset \mathbb{R}^d$ and $S(f)$ be the support of $f(x)$. The subscript on $C_0(f)$ will increase by 1 when the sampling procedure is repeated. $S(f)$ is assigned to $C_0(f)$ if and only if $S(f)$ is a bounded support of $f(x)$. If not, it is necessary to specify the boundary of $f(x)$ as $C_0(f) = S(f) \cap [a, b]^d$ where $-\infty < a < b < \infty$. Regarding the determination of constants, a and b, Wang and Fu (2007) recommend to use the presumption from recognized properties of $f(x)$ (p. 638).

3.1.2 Discretization

A descretization step is the essential procedure of the Fu-Wang algorithm. The purpose of discretization is to acquire a discrete distribution of contours (Wang & Fu, 2007, p. 638). This discretization step is a set of procedures as outlined below:

1. Forming a discrete set, $S_n(f) = \{x_j \in C_0(f), j = 1, 2, \ldots, n\}$, by a deterministic or deterministic random sequence of size n.

2. Sorting all $f(x_i)$ in descending order, e.g., $f(x_i) \geq f(x_j)$ if $i < j$.

3. Splitting $S_n(f)$ to k contours, E, with a given integer, k, such as $E_i = \{x_j : (i-1)l < j \leq il\}$ and $i = 1, 2, \ldots, k$ where $l = n/k$ and $l \in \mathbb{N}$. That is, l implies the number of points in i^{th} contour, E_i.

   Figure 3.1 facilitates the comprehension of this discretization step. An illustration in Figure 3.1 utilizes a standard distribution $f(x)$ with $k = 9$ contours.

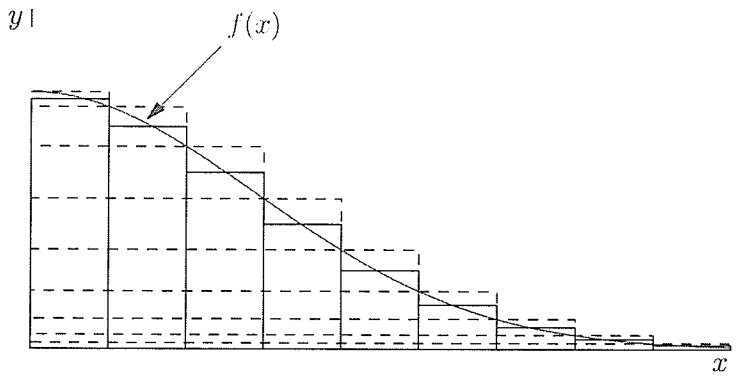


Figure 3.1: The Fu-Wang Algorithm

4. Evaluating a height of each contour E_i such that

$$\bar{f}_j = \frac{1}{l} \sum_{x_j \in E_i} f(x_j) \quad i = 1, 2, \ldots, k \tag{3.1}$$

5. Calculating a discrete distribution

$$P_k(i) = \frac{\bar{f}_i}{\sum_{j=1}^{k} \bar{f}_j} \quad i = 1, 2, \ldots, k \tag{3.2}$$

In Figure 3.1, the shaded area is quantified to acquire the discrete probability. In each contour the maximum values of $f(x^{[j]})$ and heights are indicated by dashed and solid lines respectively. Figure 3.1 clearly shows that the heights in a sequence of contours approximate $f(x)$. This has a similar behaviour of discrete probability.

3.1.3 Sampling

The sampling procedure carries out the scheme of stratified sampling mentioned in Section 2.2.2.

Assume that m random samples will be drawn.

1. Deciding a sample size of each contour proportional to the discrete probability such that $m_i = mP_k(i)$, subject to $m = \sum_i^k m_i$ and $i = 1, 2, \ldots, k$

2. Drawing m_i sample points from i^{th} contour with replacement.

3. Binding up all drawn sample points from each contour into one sequence of samples.

3.1.4 Visualizing and Updating the Significant Region

Wang and Fu (2007) emphasized the utilization of histograms on all marginal distributions of $f(x)$ to identify the appropriate significant region (p. 638). The sampling procedure will be interrupted if the identified significant region is substantially equal to C_0. If not, another sampling procedure should be conducted until the condition of significant region is satisfied.

3.2 Discussion

Even though the Fu-Wang algorithm overcomes major limitations of MCMC, it has several of its own limitations. A study by Wang and Lee (2008) has attempted to resolve limitations of the Fu-Wang algorithm. (See also the working paper entitled: "Efficient Monte Carlo Random Sample Generation through Discretization", by Wang and Lee (2008)). The emphasis placed in this working paper is related to the computational efficiency, in particular the suggestions on how to control in the low and long tail probability distribution to determine the significant region, and the establishment of the general usage of this algorithm. (See Chapter 4 for a detailed explanation of the Wang-Lee algorithm).

1. The Fu-Wang algorithm has an issue with evaluating an accurate integration under the low probabilistic region or low dimensions due to it being a sampling-

based simulation study (Fu & Wang, 2002, p. 21). It is well-known that a numerical analysis provides numerically accurate results in low dimensions rather than a sampling-based study. Once the numerical analysis is not applicable, it is preferred to use the exact probability rather than the approximate probability in a sampling-based study. In practice it is not possible to find the exact probability function from the real data. The Fu-Wang algorithm does not find the exact probability; rather it locates the approximate discrete probability. In addition, it is not an accurate boundary of the significant region on the low probabilistic region even though the significant region is identified by this sampling procedure due to its intuitive determination of boundaries. It is worthy to mention that incidences of no observations do not necessarily mean that there is no probability in the sampling-based study. Since rare samples are drawn from the low tail probabilistic area, this possibility may be overlooked in this sampling algorithm. Consequently, the imprecise construction of the discrete probability, as opposed to the representation of the true density function, arises from this loss of the potential or plausible significant region.

2. When there is no information about the properties of distribution, it is necessary to use the intuitive determination of the significant region in this algorithm. Hence, there exists the possibility of a wrong statistical inference if the shape of distribution is ill-shaped or multi-modal.

3. A study by Wang and Fu (2007) use the idea of a cut-off point on the density function $f(x^{[j]})$ to determine the significant region. This idea is well-described in Figure 3.2. Since the distribution has several modes and a long and low tail, it is difficult to identify the significant region intuitively. A loss of computational efficiency is significant if the cutoff point is too high as most of total computing time is necessarily spent on the iterative generation of random variates. The

total running time of this algorithm is measured and shown in Table 6.8, Section 6.7. Therefore, the lack of control regarding the significant region leads to the inverse relationship between cost efficiency and numerical accuracy. The imprecise discrete probability leads to the loss of the potential significant region as previously mentioned.

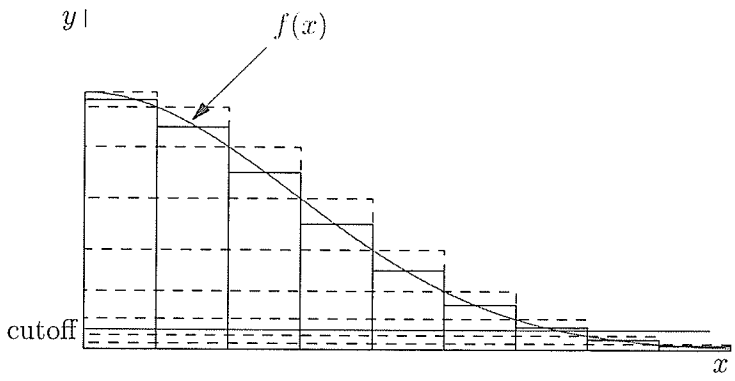


Figure 3.2: The Fu-Wang Algorithm with Cutoff

4. Currently, there is no criterion to set a proper cut-off point. Therefore, another risk exists due to the determination of a cut-off point in cases where the distribution is multi-modal and has a long and low tail probability. This results in a computational impracticality because it is only possible in theory to compute the finite number of shaded rectangles in the tail. The number of contours is fixed in advance on its implementation. Figure 3.2 shows this explanation.

5. An optimal result from the Fu-Wang algorithm is achieved by the number of contours, "a value of between 200 and 500 for a density of five dimensions or lower, and a value between 1,000 and 100,000 for a density of higher dimensions" (Wang & Fu, 2007, p. 24). It is doubtful that the number of contours is in direct proportion to the number of dimensions. As of yet, the sufficiency of computational resources and the matter of dimensionality are unresolved.

Chapter 4

Wang-Lee Algorithm

The initiative for the development of the Wang-Lee algorithm originates from the following factors: 1. cut-off point on $f(x^{[j]})$ corresponding to a significant region on $S_n(f)$, 2. control of the low probabilistic region, 3. accurate estimation of discrete probability function, and 4. computational efficiency. These factors strongly motivated the modification and development of the Fu-Wang algorithm.

The Wang-Lee algorithm consists of five procedures: initialization, discretization, contourization, two-stage sampling, and visualization. Key features of the Wang-Lee algorithm are briefly discussed here in rank order reflecting the appropriate significance to the success of the algorithm.

The appropriate sample is verified by the visualization step. Several meaningful modifications contribute to the contourization step. In particular, the Wang-Lee algorithm faciliates computational efficiency related to its unique horizontal approach for acquiring the approximate discrete probability $\{P_k(i)\}_{i=1}^{k}$. The trial method employs the cut-off point to determine the significant region by the experiment on the last partition. Moreover, a more accurate discrete probability function, which is close to the true density function, is estimated by the manipulation of the last partition regarding the large proportion of low probability. Finally, the fixed number of par-

titions regardless of the number of dimensions is suggested to estimate the discrete probability function.

4.1 Initialization

The initialization step is easily overlooked. Initialization is important as it has direct connections with the numerical accuracy of the estimates. In a Monte Carlo study, Madras (2002) made a record that "found that the accuracy of our estimate was proportional to $n^{-1/2}$, where n was the number of observations" (p. 5). In this thesis, n indicates the number of discrete base points in the Wang-Lee algorithm. In the simulation studies in Chapters 5 and 6, 1×10^7 discrete base points are employed in all illustrations and applications regardless of the number of dimensions and the model complexity.

With initialization, two computational difficulties are considered. One is the computer hardware specification and the other is the size of a vector which the statistical software provides.

The capability of computation is a significant consideration because a discrete base point in d dimensions consist of a set of d points from every coordinate, which is expressed as $x^{[j]} = (x_1, x_2, \ldots, x_d)$. A set of n discrete base points are then saved on the memory of the computer that is the RAM. When the RAM is not sufficient, the computer will use the SWAP area on the hard disk and its computing time will be immense due to the additional time of data transference between RAM and the hard disk, in addition to pure arithmetic time. This requirement necessitates a capability to store a large amount of points and an advanced programming technique to manipulate this multidimensional data matrix of n-by-d discrete base points.

The analyses of simulated data in Section 5.2.5 and genetic data in Section 6.6 by the Fu-Wang algorithm demonstrate this immense computing time. Both of these

examples have 24 parameters into which the discrete base points need to be allocated. Hence, there is a substantial memory requirement of $1.92GB$ for storing discrete base points and manipulating 10^7-by-24 data matrix efficiently.

This feature exemplifies the substantial advantages of the Wang-Lee algorithm, primarily, the unrestricted dimensionality. The Wang-Lee algorithm manipulates only indices of array (or data matrix) for the sampling procedure. One consideration of this manipulation is that the maximum length of a vector up to 1 or 2×10^9 For this specification see Memory Limits in R, R Development Core Team (2008a). Hence, the stipulation that advanced programming techniques and skills are required to manipulate total discrete base points with this vector system.

Regarding the initialization of sample size, $m = 10^3$ is optimal to characterize the density function $f(x)$ by the repetitions of simulations.

Algorithm 4.1.1 Initialization

Require:

$f(x)$ a given or target density function with an analytical form.

$m \leftarrow$ sample size

$n \leftarrow$ discrete base points

$k \leftarrow$ partitions

$last.p \leftarrow$ weight on the last partition

4.2 Discretization

The purpose of this discretization procedure is to obtain the dense discretized sampling space $S_n(f)$ which contains the significant region of $f(x)$. This sampling space $S_n(f)$ is used to query indice of a sample by the utilization of approximate discrete probability $\{P_k(i)\}_{i=1}^k$ later. This idea is clearly linked to the concept of importance sampling as shown in Section 2.2.1. A good sampling necessitates that

samples concentrate on the significant region of the density function $f(x)$ without regard to the negligible region. The acquisition of a dense and highly concentrated sampling region is crucial in the accurate estimation of density function $f(x)$.

This discretization procedure of the Wang-Lee algorithm is identical to the initial compact cover step of the Fu-Wang algorithm in Section 3.1.1. The strategy of how to construct the compact cover is explained with more detail as follows: In a given density function $f(x)$, assume that $C_0(f)$ is the initial compact set subject to $C_0 \subset \mathbb{R}^d$ and $S(f)$ is the support of $f(x)$.

1. Set $C_0(f) = S(f)$ if $f(x)$ has a bounded support.

2. Determine the constants a_i and b_i where $-\infty < a < b < \infty$ for $i = 1, 2, \ldots, d$ with d dimensions if $f(x)$ does not have a bounded support. Then, set $C_0(f) = S(f) \cap [a, b]^d$.

3. Ensure that the significant region of $f(x)$ is the subset of $C_0(f)$.

In practice, it is difficult to be aware of the properties of $f(x)$ so that the determination of constants (a and b in i^{th} dimension which contains the significant region) is arbitrary work. This limitation is discussed in Section 3.2 when $f(x)$ is ill-shaped and multi-modal.

Subsequent to the specification of the safe boundaries on the sampling space $S(f)$, the Wang-Lee algorithm generates and accumulates all discrete base points $x^{[j]}$ after trimming all zero values of $f(x^{[j]})$, where $j = 1, 2, \ldots, n$. This accumulation of discrete base points is repeated until a given initial number of n discrete base points is obtained. All discrete base points $x^{[j]}$ are, then, eliminated if $f(x^{[j]}) < 5 \times 10^{-324}$ due to the specification of GNU R, R treats the double type of float values less than 5×10^{-324} as 0 For this specification see Numerical Characteristics of the Machine R Development Core Team (2008b). The discrete base points will be generated as equals as the number of discrete base points are eliminated. This procedure contributes to the

computational speed of the Wang-Lee algorithm when compared to the acquisition of a dense and highly concentrated sampling space by the Fu-Wang algorithm.

Algorithm 4.2.1 Discretizing $S(f)$ to $S_n(f)$

while $nrow(f(X)) = n$ **do**

Generation Step

for $i = 1$ to d **do**

$\underset{n\times i}{X} \leftarrow \underset{n\times 1_i}{\mathbf{x}} \sim U(a_i, b_i)$

end for

Evaluating $f(\underset{n\times d}{X})$

if $\underset{n\times 1}{f(X)} > 0$ **then**

Save the indice

else

Repeate Generation Step recursively

end if

end while

$\underset{n\times(d+1)}{X} \leftarrow \underset{n\times d}{X} + \underset{n\times 1}{f(X)}$

return $S_n(f) = \underset{n\times(d+1)}{X}$

4.3 Contourization

The contourization step is the core of the Wang-Lee algorithm. The object of contourization is to get possession of the approximate discrete probability for the next sampling procedure. Wang and Fu (2007) note two properties of the contourization: 1. an intermediate tool to identify the significant region by transforming $f(x)$ into a monotone discrete probability, and 2. provide information about the approximate modes by the first contour (p. 639). The Wang-Lee algorithm appropriates these properties from the Fu-Wang algorithm. The modification of this contouriza-

tion procedure derives computational efficiency and shows the possibility of how to control the significant region. This contourization procedure consists of two steps, the partitioning and the approximate discrete probability.

4.3.1 Partitioning

The Wang-Lee algorithm conveys greater efficiency via the horizontal approach for partitioning on the discretized sample space $S_n(f)$ than the Fu-Wang algorithm does.

Estimating the target density function $f(x)$ utilizing the Fu-Wang algorithm is considered as a vertical approach on the discretized sample space $S_n(f)$. The Fu-Wang algorithm initially divides the discretized sample space $S_n(f)$ into equally-spaced k contours, and then calculates the level of each contour by the utilization of equal number of points l in the ith contour as described in Section 3.1.2. This algorithmic procedure is clearly shown in Figure 3.1 and described in Section 3.1.2.

In contrast, the Wang-Lee algorithm employs a horizontal approach on the discrete sample space $S_n(f)$ and only values of $f(x^{[j]})$. First, the Wang-Lee algorithm saves the indices of sorted $f(x^{[j]})$ in descending order. This small indice technique contributes to the computational efficiency for querying the discrete points as the final samples later.

The Wang-Lee algorithm then establishes levels of k partitions to determine the domain of k partitions on the discretized sample space $S_n(f)$. The recommended levels of k partitions are chosen as the mid-point in k equi-distant vertical interval on the range of $f(x^{[j]})$ for computational convenience. This functional distinction equates to the elimination of a procedure from the Fu-Wang algorithm — the calculation of levels of k contours. In essence, equation (3.1) in Section 3.1.2 is removed from the Fu-Wang algorithm.

Figure 4.1 is displayed here for convenience of the reader and to illustrate with

clarity the differences and subsequent advantages associated with this approach to partitioning. An illustration in Figure 4.1 applies the standard normal distribution $f(x)$ with $k = 9$ partitions as the as Figure 3.1. The different directional approach of the Wang-Lee is clearly distinguished in Figures 4.1 compared to Figure 3.1 in Section 3.1.2.

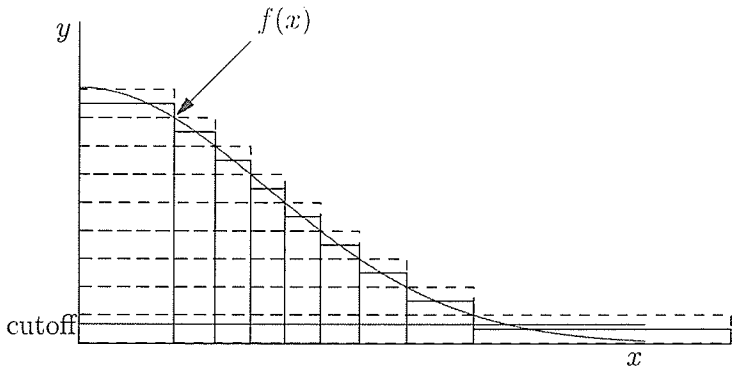


Figure 4.1: The Wang-Lee Algorithm

In Figure 4.1, the dashed lines imply that the equally divided intervals on the range of $f(x^{[j]})$. The solid lines indicate the level of i^{th} partition, which is set as the mid point in the i^{th} interval, and specifies the area of ith partition on the discretized sample space $S_n(f)$. The shaded area under the solid line will be employed to obtain the approximate discrete probability.

A nature of the directional shift for contourizing the discretized sample space $S_n(f)$ contains the different number of discrete base points $x^{[j]}$ in each partition $E_i(x^{[j]})$. In contrast, the Fu-Wang algorithm has the equal number of l discrete base points $x^{[j]}$ over all partitions. (See also procedure 3 in Section 3.1.2). Then, the k partitions on the discretized sample space $S_n(f)$ are dynamically determined respectively corresponding to the levels of k partitions such that

$$h_i < E_i(x^{[j]}) < h_{i+1} \tag{4.1}$$

where h_i is the level of i^{th} partition established by the mid-point of every interval on the range of $f(x^{[j]})$. This approach suggests the tentative criterion regarding the cut-off point on $f(x^{[j]})$ later — the level of last partition — to manage the significant region on the discretized sample space $S_n(f)$ by the last partition.

Controversy exists in the setting of the cut-off point on $f(x^{[j]})$ since the value of cut-off point is in direct proportion to the length of the range of $f(x^{[j]})$; that is, the length between maximum and minimum of $f(x^{[j]})$. This will be explained in greater depth in Section 4.3.2.

It is clearly evident that the number of $f(x^{[j]})$ between the successive levels of k partitions is equivalent to the number of discrete base points in each partition $E_i(x^{[j]})$. This feature leads to another computational efficiency. Ergo a function hist() in R (or different function names on other statistical software) performs both tasks simultaneously: building the levels of k partitions and counting the number of $f(x^{[j]})$ on every partition.

In simulation studies in Chapters 5 and 6, the number of partitions, $k = 10^5$, is specified. A large proportion of partitions returns to zero value. This is significant to interpreting the shape of distribution $f(x)$ and the location of high probabilistic region since the information of shape about the target density function $f(x)$ is carried into a sequence of $E_i(x^{[j]})$ by this contourization. That the partitions have zero values of numbers of discrete base points $x^{[j]}$ is an indication that the values of a density are dropping rapidly or that the region is negligible for the sampling. Figure 4.1 facilitates comprehension of the interpretation of distribution shape.

The contourization procedure is summarized as follows:

1. Save the indices of sorted $f(x^{[j]})$ in descending order.

2. Set the level h_i of i^{th} partition as the mid point in equi-distant occurrences on the range of $f(x^{[j]})$

3. Establish the domain of k partitions on a discretized sample space $S_n(f)$ such that $E_i = \{x^{[j]} : h_i < f(x^{[j]}) \leq h_{i+1}\}$

4. Count the the number of $f(x^{[j]})$ in the ith partition E_i

Algorithm 4.3.1 Contourization with k partitions

for $j = 1$ to n **do**

  if $j' > j$ **then**

    $f(x^{[j']}) \geq f(x^{[j]})$

  end if

end for

for $i = 1$ to k **do**

  $h_i = \sum_{i=1}^{k} \frac{(\max f(x^{[j]}) - \min f(x^{[j]}))}{k}$

  $n_i = \mu^*(\tilde{E}_i) = \{x^{[j]} : h_{i-1} < f(x^{[j]}) \leq h_i\}$

end for

return $n_i = n(E_i)$

4.3.2 Approximate Discrete Probability Function

The approximate discrete probability $\{P_i(k)\}_{i=1}^{k}$ in the Fu-Wang algorithm is defined as the equation (3.2) in Section 3.1.2. In the Wang-Lee algorithm, the approximate discrete distribution is implemented in a different way as follows:

$$\begin{aligned} P_k(i) &= \frac{h_i n_i}{\sum_{j=1}^{k} h_i n_i}, \quad i = 1, 2, \ldots, k-1 \\ P_k(i) &= w\frac{h_i n_i}{\sum_{j=1}^{k} h_i n_i}, \quad i = k, \quad 0 < w < 1 \end{aligned} \tag{4.2}$$

where n_i is the number of $f(x^{[j]})$ and h_i is the level of i^{th} partition, $i = 1, 2, \ldots, k$, respectively, and $w = 0.5$ is a standard weight.

The superiority of the Wang-Lee algorithm arises from the appropriate weight on the last partition due to the consideration of low probability in the tail of distribution $f(x^{[j]})$. Typically, the last partition is overestimated as much as the difference between the area A and the shaded area B in Figure 4.2. This results from the establishment of a standard unit rectangle for the approximation.

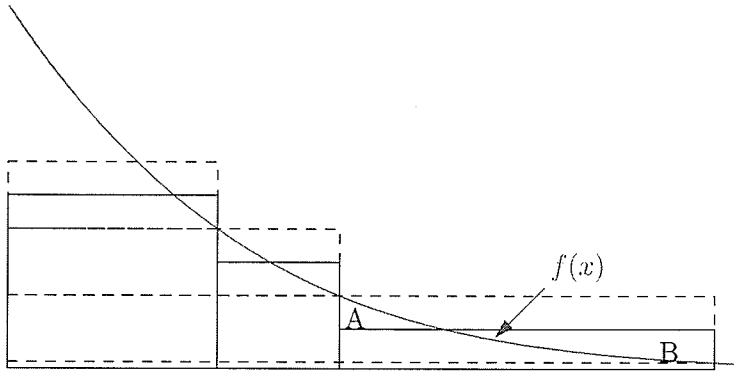


Figure 4.2: Overestimation Probability on Last Partition

The illustration in Figure 4.3 clearly shows the impact of the overestimation and its control on the last partition E_k.

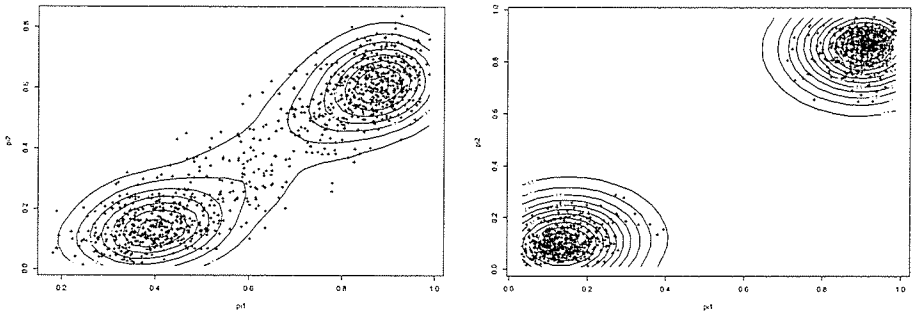


Figure 4.3: Overestimation and After Controlled

The left panel in Figure 4.3 represents the uncontrolled case of probability on the last partition E_k. It is clearly evident that inappropriate samples are drawn from the

negligible area. This phenomenon is apparent when the cut-off point on $f(x^{[j]})$ in the Fu-Wang algorithm is established too low in a mixture model or a hierarchical model. (See also Figure 5.6 in Section 5.3). The necessity of manipulation on the probability of the last partition is clear.

Currently, the Wang-Lee algorithm sets the weight $w = 0.5$ on the probability of the last contour E_k as an experiment for further research. In considering about the weight, the quantity could be approximately close to the half of the last unit rectangle (as equivalent as the area of B−A in Figure 4.2). The quantity of weight is tested for all of the examples and applications in Chapters 5 and 6. Some examples require manipulation regarding the weight to draw the appropriate samples. In Section 5.3, a practical guide will illustrate the selection of appropriate weight on the last partition and its consequence.

Algorithm 4.3.2 Approximate discrete probability function $P_i(k)$

for $i = 1$ to k **do**

$p_i(\tilde{E}_i) = \frac{h_i n_i}{\sum_{i=1}^{k} h_i n_i}$

end for

$p_k \leftarrow w \times p_k$

for $i = 1$ to k **do**

$P_i(E_i) = \frac{p_i(E_i)}{\sum_{i=1}^{k} p_i(E_i)}$

end for

return CDF of $\{P_i(E_i)\}_{i=1}^{k}$

4.4 Two-Stage Sampling

The Wang-Lee algorithm follows the identical sampling procedure as the Fu-Wang algorithm. Assume that m random samples are drawn. Since a partition E_i is regarded as a stratum in the Wang-Lee algorithm, $\cup_{i=1}^{k} E_i = S_n(f)$ and $E_i \cap E_j = \emptyset$ for

$\forall i \neq j, i = 1, 2, \ldots, k$.

1. Deciding a sample size of each contour proportional to the discrete probability such that $m_i = mP_k(i)$, subject to $m = \sum_i^k m_i$ and $i = 1, 2, \ldots, k$
2. Drawing m_i sample points from i^{th} partition with replacement.
3. Binding up all drawn sample points from each partition into one sequence of samples.

The Fu-Wang algorithm and the Wang-Lee algorithm have the same foundation for this sampling procedure. However, the inverse CDF technique to query the indice of discrete base points $x^{[j]}$ is implemented by the utilization of complete vectorization technique in the Wang-Lee algorithm.

Algorithm 4.4.1 Two-Stage sampling

for $i = 1$ to k **do**
  for $j = 1$ to n_i **do**
    $m_i \sim \mathcal{U}(0, 1)$
    Querying m_i from n_i
  end for
end for
$m = \sum m_i$
return a sample of size m

4.5 Visualization

The utilization of all marginal distribution of $f(x)$ for all dimensions is the effective way of diagnosing whether or not the determination of appropriate samples is completed from the significant region as described in Section 3.1.4.

When the significant region of $f(x)$ is well identified within the initial compact support $C_0(f)$, the drawn sample is appropriate for use in the inference. However, the adjustment on the initial compact support C_0 will be required if the significant region is the subset of the initial compact support $C_0(f)$. The technical use of visualization is described for the use of graphical diagnostic in Section 5.3.

4.6 R Module Implementation

The next R program code is a generalization of the Wang-Lee algorithm.

```
# A General Procedure in Wang-Lee Algorithm

f.name <- function(

 # Initialization
 n.discrete.pnts=1e7,  # Number of discrete base points
 n.cnts=1e5,    # Number of partitions
 n=1e3,     # Number of samples
 last.weight=0.5, # Weight on last contour
 ){

        # Generating base discrete points in the significant region
        vars  <- runif(n.discrete.pnts, 0, 1);

        density <- # Defining the target distribution

        # Creating D-dimeional discretized compact sample space
        sample.space <- data.frame(var_1[, var_2, ...] , density=density
            );

        # Trimming zeros
        density <- density[which(density>0)];

# Sorting by decending order
density.ind <- order(density, decreasing=TRUE);

# Contourization
lebergue.measure <- hist(density, breaks=seq(from=min(density), to=max(
    density), length.out=n.cnts+1), plot=FALSE);

# Discrete Probabilities on partitions
get.pdf <- rev(lebergue.measure$counts * lebergue.measure$mids);

# Weighting the last partition
get.pdf[n.cnts] <- last.weight*get.pdf[n.cnts];

```

```
35  # Normalizing CDF
36  nmlzd.cdf <- c(0, cumsum(get.pdf)/sum(get.pdf));
37
38  # Sampling by inverse CDF index searching
39  rnd.variates = runif(n);
40  p.sample.size <- hist(rnd.variates, breaks=nmlzd.cdf, plot=FALSE)$
        counts;
41  valid.samples <- which(p.sample.size>0);
42  sample.cnt <- mapply(sample, MoreArgs=list(replace=TRUE), rev(lebergue.
        measure$counts), p.sample.size);
43  cum.pnts.ind <- c(0, cumsum(rev(lebergue.measure$counts)));
44  cum.pnts.ind <- cum.pnts.ind[-(n.cnts+1)];
45  sample.list <- mapply("+", as.list(cum.pnts.ind), sample.cnt);
46  sample.ind <- unlist(sample.list);
47
48  # Saving Final samples
49  sample.space <- sample.space[density.ind[sample.ind],];
50 }
51
52 # Simulation
53 f.name(n.discrete.pnts=1e7, n.cnts=1e5, n=1e3);
```

Chapter 5

Simulation Study

5.1 Simulation Plan

This chapter features various attributes of the Wang-Lee algorithm which are interpreted in the context of the Wang-Lee algorithm using representative distributions. For the convenience of the reader, many of the illustrations in this chapter are derived from Fu and Wang 2002 study. Other source illustrations are classic examples from recent literature which challenges the limitations of MCMC. Each representative illustration has been selected with care for the express purpose of comprehensively delineating the unique features of the Wang-Lee algorithm as follows: 1. procedural description, 2. numerical accuracy, 3. identification of multimodes, 4. freedom from the need for reparameterization or additional adaptation, and 5. computational efficiency in high dimensions. A practical guide for a graphical diagnostic is also provided to facilitate greater insight and furnish details pertinent to this methodology.

Controlling of the simulation parameters is substantiated using examples on a case-by-case basis. The default parameters on the simulation in Chapter 5 and Chapter 6 are as follows:

1. 1×10^7 discrete base points, n.

2. 1×10^5 partitions, k.

3. 1×10^3 samples, m.

4. Weight on the last partition, $w = 0.5$.

All programs in this simulation are written in GNU software, R-2.6.2, provided by R Development Core Team (2008c), and subsequently tested on the Linux (Ubuntu 8.04, Hardy) system with GCC 4.2.4 (x86_64-linux-gnu). The hardware specifications are as follows: Dual Core AMD Opteron(tm) Processor 275 (2193.745 MHz), 7992 MiB Memory with SWAP 3153 MiB, and 1024 KB L2 Cache stystem.

5.2 Case Studies

5.2.1 Bivariate Beta Distribution

This example will incontrovertibly exhibit the operation of the Wang-Lee algorithm; its numerical accuracy and its graphical coincidence between the theoretical results and simulation. For this purpose, a simple bivariate beta distribution of two independent random variables is studied such that $X_1 \sim Be(2,2)$ and $X_2 \sim Be(3,1)$ up to normalizing constant as equation 5.1 (Fu & Wang, 2002, p. 13).

$$f(x_1, x_2) = x_1(1 - x_1)x_2^2, \;\; 0 \leq X_1, X_2 \leq 1 \tag{5.1}$$

In obtaining a sample of size $m = 1 \times 10^3$, it is critical to ascertain the sampling space $S_n(f)$. As most of the samples will be obtained from the significant region, a failure to appropriately identify the sampling region results in an invalid sample. In this example, the information regarding the sampling space is presently provided as $[0,1] \times [0,1]$. This is a case of compact support; the initial compact support being specified as $C_0(f) = S_n(f) = [0,1] \times [0,1]$. Then, $n = 1 \times 10^7$ discrete base

points are generated to discretize the initial compact support, $C_0(f)$. $k = 1 \times 10^5$ partitions are used to contourize the evaluated density function, $f(x^{[j]})$. That is, the discretized sampling space $S_n(f)$ is sectionalized with k partitions corresponding to the midpoint in each equi-distant interval on the range of $f(x^{[j]})$. These k partitions distingusish the configuration of the monotone discrete distribution and the domain of discretized sampling space. The weight on the last partition w is set by 0.5 to eliminate the consequences inherent to sampling from the negligible region. The question of weight (as previously discussed in Section 4.3.2) will be demonstrated by the empirical approach in Section 5.3. Subsequent to the establishment of the discrete probability function is the two-stage sampling procedure.

The first two sample moments are investigated and contrasted with the values found by the Fu-Wang algorithm. The numerical result from the Wang-Lee algorithm closely approximates the theoretical standards. These results are listed in Table 5.1.

Table 5.1: Mean and Standard Deviation of Equation (5.1)

Statistics	$E(X_1)$	$E(X_2)$	$SD(X_1)$	$SD(X_2)$
Theory	0.0500	0.7500	0.2236	0.1936
Fu-Wang	0.5073	0.7572	0.2258	0.2001
Wang-Lee	0.4922	0.7448	0.2149	0.1985

Investigation of the marginal histograms of X_1 and X_2 in Figure 5.1 validates whether or not the significant region is contained within the initial compact support C_0. Thus, sufficient evidence prevails to establish that no adjustment is required on the inital compact support C_0. A scatter plot, a surface plot, and two histograms of the marginal distributions of two random variables X_1 and X_2 are provided in Figure 5.1. On the histograms, the solid and dashed lines indicate the fitted line from the drawn sample and the theoretical density curve respectively. By a graphical and

numerical comparison, it is evident that the Wang-Lee algorithm renders the relevant sample for the two dimensional beta distribution.

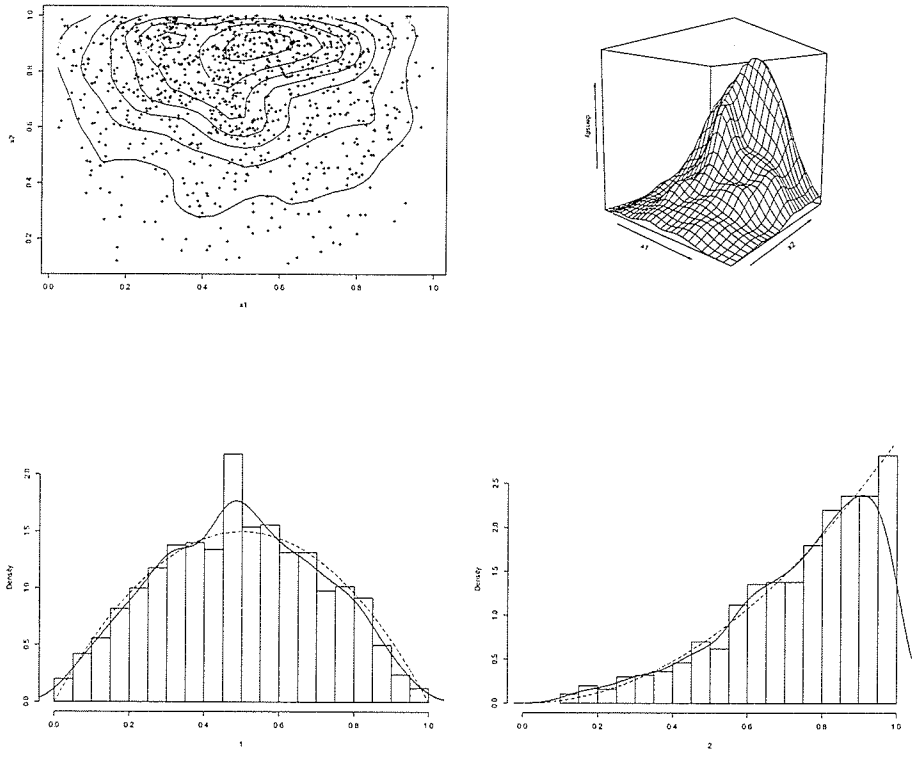


Figure 5.1: Scatter, Contour Plot, and Histograms of Equation (5.1)

5.2.2 Dirichlet Distribution

A dirichlet distribution of three random variables, $(X_1, X_2, X_3) \sim D(0.5, 2.5, 4.5, 6.5)$, is found from Fu and Wang's 2002 study, written in the analytic form up to the normalizing constant with $0 \leq x_1 + x_2 + x_3 \leq 1$ (p. 14).

$$f(x_1, x_2, x_3) = x_1^{-0.5} x_2^{1.5} x_3^{3.5} (1 - x_1 - x_2 - x_3)^{5.5} \tag{5.2}$$

The Wang-Lee algorithm unequivocally secures the pertinent sample from this multi-dimensional dirichlet distribution. The drawn sample from this distribution is exam-

ined for the numerical accuracy and graphical coincidence between theorical results and simulation. Since the marginal distribution of the multidimensional dirichlet distribution is the beta distribution, the simulation result can be contrasted. The theoretical marginal distrbutions are found as $X_1 \sim Be\ (0.5, 13.5)$, $X_2 \sim Be\ (2.5, 11.5)$, $X_3 \sim Be(4.5, 9.5)$. Two sample moments from both the simulation and theory are compared in Table 5.2.

Table 5.2: Mean and Variance of Equation (5.2)

	MEAN			VARIANCE		
	Theory	Fu-Wang	Wang-Lee	Theory	Fu-Wang	Wang-Lee
X_1	0.0357	0.0352	0.0359	0.0022	0.0022	0.0023
X_2	0.1785	0.1749	0.1725	0.0097	0.0095	0.0096
X_3	0.3214	0.3169	0.3203	0.0145	0.0160	0.0160

The graphical coincidence between the theoretical density curve and the fitted line in the a sample is represented in Figure 5.2. On the histograms, the solid and dashed lines indicate the fitted line from the drawn sample and the theoretical density curve respectively. As the significant sampling region is located within the initial compact support, no modification to the initial compact support C_0 is required. Thorough examination of the graphical diagnostic and numerical comparison establish the validity of the Wang-Lee algorithm to originate the relevant sample from the multidimensional dirichlet distribution.

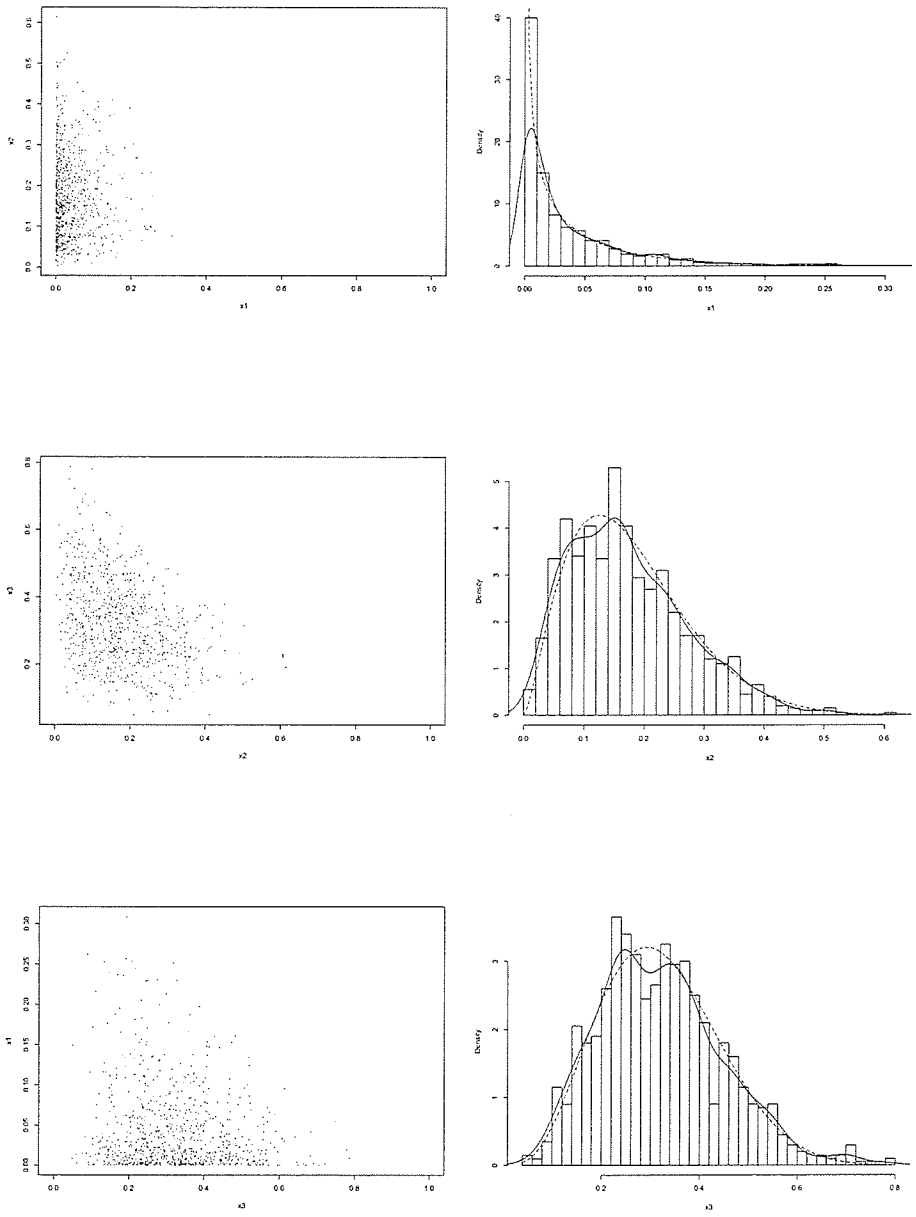


Figure 5.2: Scatter Plot and Histograms of Equation (5.2)

5.2.3 A Bimodal Example

In 1993, West provided a density of two parameters π_1 and π_2

$$(\pi_1(1-\pi_2))^r(\pi_2(1-\pi_1))^s(1-\pi_1(1-\pi_2)-\pi_2(1-\pi_1))^{(n-r-s)} \tag{5.3}$$

with $n = 45, r = 5, s = 3$ and $0 < \pi_1, \pi_2 < 1$ and, employing this equation 5.3 to demonstrate the deficiency of MCMC known as — "mixture collapsing" (p. 414) — attributable to the large proportion of low probabilities between high densities.

The Wang-Lee algorithm is an innovative solution to this type of density function. A contour plot integrated with a scatter plot and histograms of the marginal distributions of parameters π_1 and π_2 are provided in Figure 5.3. (See also Fig. 1. for the exact contours and weights in West, 1993, p. 415; Figure 1. in Oehlert, 1998, p. 165).

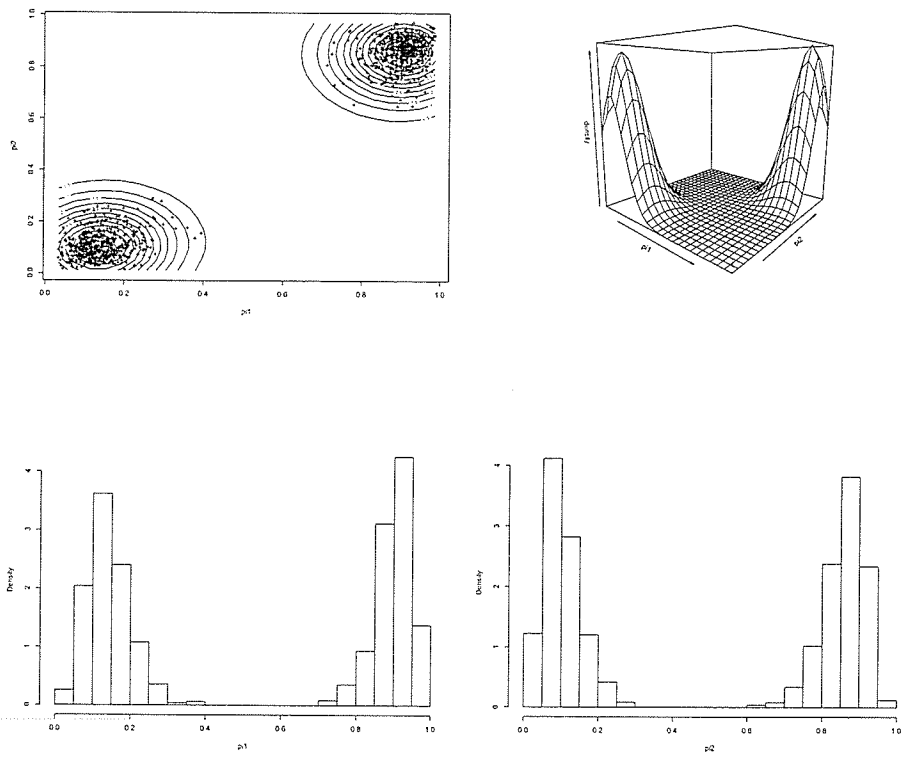


Figure 5.3: Scatter, Contour Plot, and Histograms of Equation (5.3)

Monitoring several modes remains the major impediment associated with the MCMC algorithm as pointed out in Section 1.2. This can be imputed to the incapacity of MCMC to identfy the complete range of the posterior mode (Celeux et al., 2000, p. 957). This capacity to capture the modes by the Wang-Lee algorithm is significant for data exploration. Confirmation has been verified by repeated simulations on a mixture model of various numbers of components with diverse variances and correlations. The expanded example is illustrated in Section 5.2.4.

The Wang-Lee algorithm provides supplementary numeric information relative to the approximate mode. This attribute is derived by the contourization procedure described in Section 4.3. This information is expedient to the approximation of the maximum value of density function when it is not evident from the graphical analysis as in the histograms in Figure (5.3). The information of approximate modes is listed in Table 5.3.

Table 5.3: Mean, Standard Deviation, and Modes of Equation (5.3)

Statistics	Fu-Wang		Wang-Lee	
Mean	0.5107	0.4672	0.5264	0.4858
SD	0.3805	0.3825	0.3820	0.3808
MODE	0.9258	0.8804	0.9225	0.8803
	0.1178	0.0764	0.1191	0.0741

5.2.4 A Multimodal Example

The previous three examples illustrated the validity of the Wang-Lee algorithm in rendering the appropriate sample in low dimensions with the compact support. Section 5.2.4. will illustrate the efficient utilization of the Wang-Lee algorithm on the unbounded support.

A Gaussian mixture model as equation (5.4) is frequently used to demonstrate

various adaptive MCMC method to reveal the deficiencies of the standard MCMC (See Fu & Wang, 2002, pp. 16–17; W. R. Gilks, Roberts, & Sahu, 1998, p. 1051; and Liang, Liu, & Carroll, 2007, p. 311).

$$\begin{aligned} p(\mathbf{x}) =& \frac{1}{3}N\left[\begin{pmatrix} -8 \\ -8 \end{pmatrix}, \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}\right] \\ &+ \frac{1}{3}N\left[\begin{pmatrix} 6 \\ 6 \end{pmatrix}, \begin{pmatrix} 1 & -0.9 \\ -0.9 & 1 \end{pmatrix}\right] + \frac{1}{3}N\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right] \end{aligned} \tag{5.4}$$

The parameters employed in this illustration are identical to those specified in the study of Liang et al. (2007)(p. 311).

The initial compact support $C_0(f)$ will be designated by information regarding mean and variance from each mixture component as no other information on the support in this model exists. Since the first and second components have $\mu = (-8, -8)$ and $\mu = (6, 6)$ respectively and its spread are both $(1, 1)$, the sampling space $S(f)$ is determined as $\mu_1 - \times\sigma_1 = -8 - 3 \times 1 = -11$ and $\mu_2 + 3 \times \sigma_2 = 6 + 3 \times 1 = 9$. Therefore, initial compact support is $C_0(f) = [-11, 9]$ and proceeds to the sampling procedure.

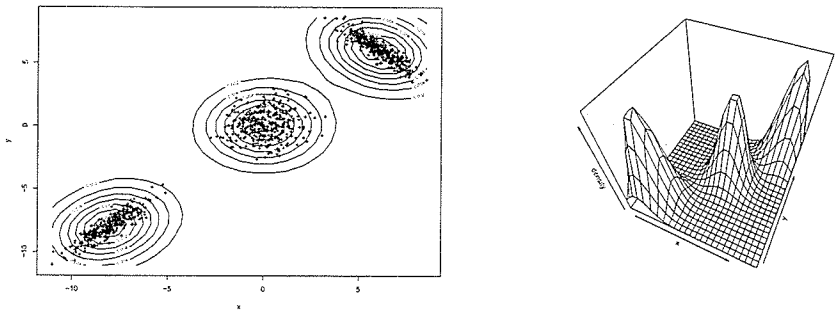
Subsequent to sampling on the initial compact support, the analysis of histograms of all marginal distributions signify that the adjustment on the intial compact support is essential as the significant region is not a subset of C_0. It is revised as $C_t(f) = [-12, 11] \times [-12, 11]$ after t times of repetitions. These repetitions are to be continued until such time as no further graphical nor numerical changes are found.

The mean, standard deviation, and approximated modes of a mixture are provided in Table 5.4.

Table 5.4: Mean and Standard Deviation of Equation (5.4)

Statistics	Fu-Wang		Wang-Lee	
MEAN	-0.7010	-0.7831	-0.6237	-0.6386
SD	5.8425	5.7940	5.8929	5.8664
MODE	-8.0378	-8.0457	-7.9717	-7.9559
	6.0460	5.9658	6.0130	5.9598

A contour plot, a scatter plot and histograms of all marginal distributions are shown in Figure 5.4 (See also Figure 3 Contour plots in Liang et al., 2007, p. 312). Consequently, it is unequivocally illustrated that a drawn sample by the Wang-Lee algorithm is appropriate to project this Gaussian mixture model. As well, the redundancy of reparameterization or proper adaptation in the Wang-Lee algorithm is illustrated.



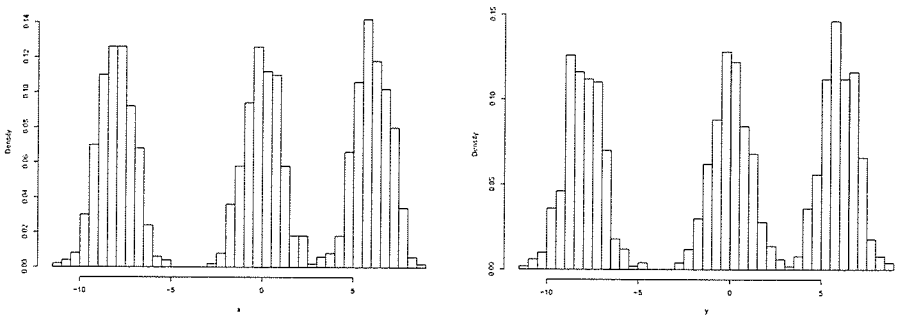


Figure 5.4: Scatter, Contour Plot and Histograms of Equation (5.4)

5.2.5 Simulated Data

The example in this section illustrates the computational efficiency and precision of the numerical approximation of the Wang-Lee algorithm in high dimensions.

This example was originally used to test the congruity of the Fu-Wang algorithm and examine the SLC190 genetic data set (See Wang & Fu, 2007, pp. 643–645). The analysis of the SLC190 Genetic data will be illustrated in Section 6.6.

In this data analysis, the Bayesian hierarchical mixture model is utilized with an unknown number of sub-populations identical to the equation (1.1) in Section 1.1.1.

As established in Section 1.1.1 and 1.1.2, the model complexity and analytically non-closed form of integration results in the failure of the standard MCMC algorithm. Section 5.2.5 will substantiate that the Wang-Lee algorithm definitively resolves the mathematical and computational challenges of the standard MCMC algorithm.

The presumptions of the simulated data and model specifications are identical to Wang and Fu's 2007 study (See Wang & Fu, 2007, p. 643).

The simulated data is a random sample of size of $N = 200$ from the mixture of $\mathcal{N}_1(\mu_1 = 3, \sigma^2 = 1)$, $\mathcal{N}_2(\mu_2 = 6, \sigma^2 = 1)$, and $\mathcal{N}_3(\mu = 9, \sigma^2 = 1)$ with the probabilities, $w_1 = 0.64$, $w_2 = 0.32$, and $w_3 = 0.04$ respectively.

The density function of the mixture of K sub-populations is designed as identical

to the equation (1.1) in Section 1.1.1. Equal variances of all sub-populations are assumed. Prior information is synonymously applied such that $\mu^K \sim \mathcal{N}(\mu_0, \sigma_0^2)$, $\sigma_K^2 \sim \mathcal{IG}(\alpha, \beta)$, and $w^K \sim \mathcal{D}(\gamma)$ with $\alpha = 2$, $\beta = (R_y/6)^2$, and $\gamma = 1$.

The likelihood function and full posterior density are expressed as the equation 1.2 and the equation 1.4 respectively in Section 1.1.2.

When the number of sub-population is set by $K_{\max} = 4$, the number of parameters to be estimated is 21. The applied specification of compact support is $\mu^K \in [0, 12]^K$, $\sigma_K^2 \in [0.1, 5]^K$ and $w^K \in [0, 1]^K$.

The significant region is verified by the repeatd simulations with manipulation of the weight on the last partition until the explicit and appropriate histograms of marginal distributions are achieved and no further modifications to the numerical results. The appropriate weight is found as $w = 0.01$ in this example.

Table 5.5 and 5.6 include the estimated marginal posterior distribution with regard to the number of component K and approximated posterior modes, means and standard deviations are provided respectively.

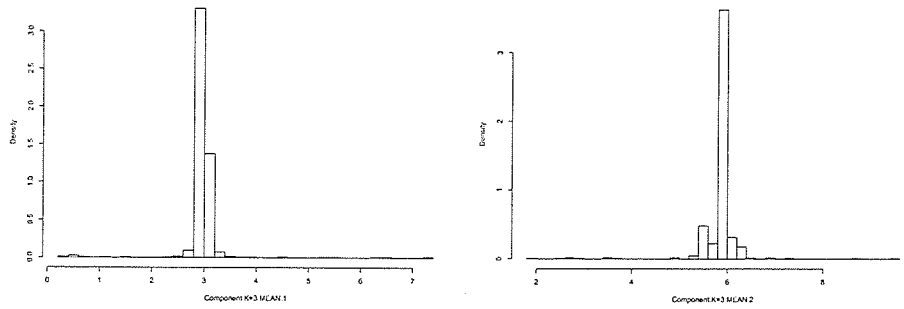
Table 5.5: Prior and Posterior distribution of K from Equation (1.4)

K	1	2	3	4
Prior	0.25	0.25	0.25	0.25
Fu-Wang	0.0100	0.0120	0.5500	0.4280
Wang-Lee	0.0170	0.0260	0.6660	0.2910

Table 5.6: Parameters, Posterior Means and Standard Deviations of μ^3 and w^3 for $K = 3$ from the Equation (1.4)

$K = 3$	Fu-Wang		Wang-Lee	
TRUE	MEAN	SD	MEAN	SD
$\mu_1^3 = 3$	3.0205	0.1668	2.9952	0.3873
$\mu_2^3 = 6$	5.8742	0.2447	5.8802	0.4206
$\mu_3^3 = 9$	9.5504	0.6554	9.6219	0.6758
$w_1^3 = 0.64$	0.5970	0.0511	0.5848	0.0704
$w_2^3 = 0.32$	0.3620	0.0507	0.3659	0.0542
$w_3^3 = 0.04$	0.0408	0.0214	0.0491	0.0671
σ^3	1.0808	0.2049	1.0972	0.4487

Histograms of the marginal distributions are listed in Figure 5.5 (See also Figure 6. in Wang & Fu, 2007, p. 645).



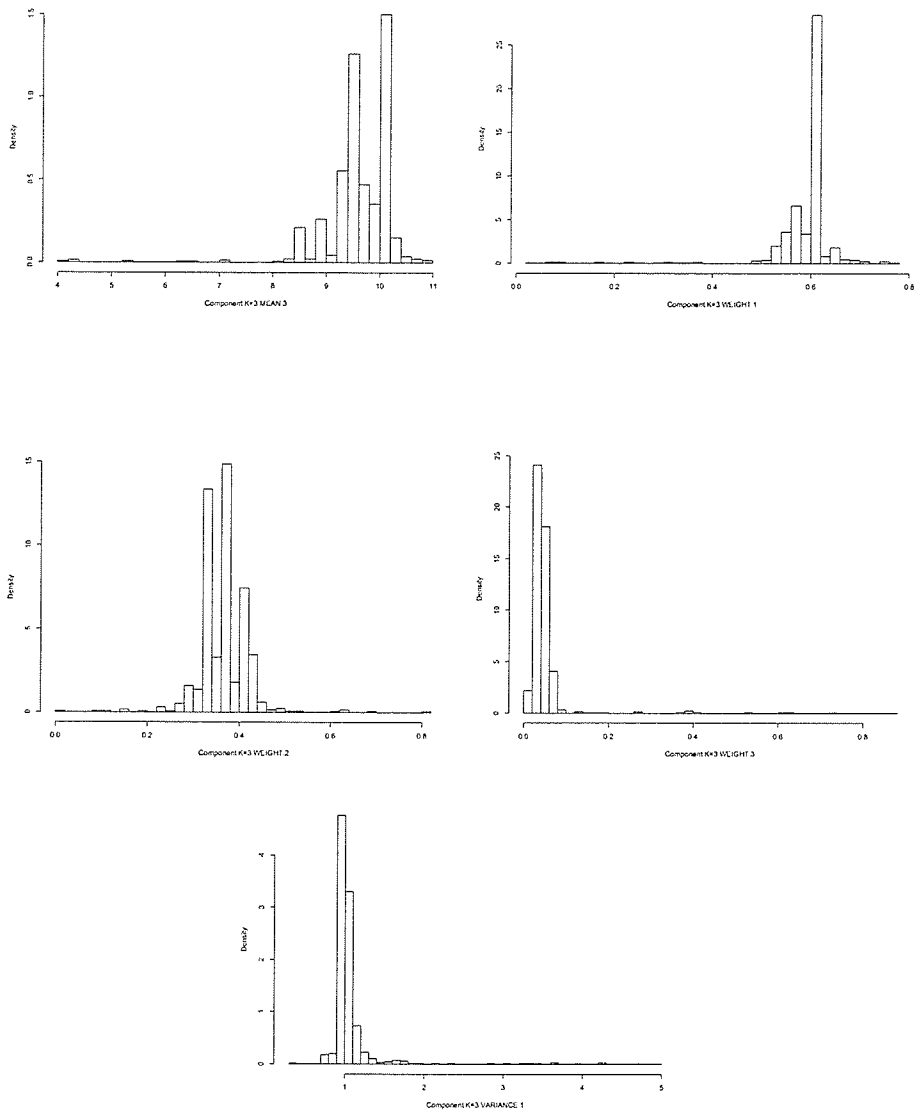


Figure 5.5: Histograms of $\mu_i^{K=3}$, $\sigma^{K=3}$, and $w_i^{K=3}$ from Equation (1.4)

High computational efficiency is achieved using the Wang-Lee algorithm rather than the Fu-Wang algorithm. Comparison of the computing time is provided in Table 5.7, Section 5.2.6. The Wang-Lee algorithm shows a computational time that is five times faster than the Fu-Wang algorithm in 21 dimensions of Bayesian hierarchical model.

5.2.6 Computing Times

Running time of each algorithm is measured by proc.time() function in R. It provides the user, system, and total time from the computing system (See the definitions and specifications in Running Time of R in R Development Core Team, 2008d).

In computations wherein the dimension is less than four, it is apparent that the difference in computing time is minimal. The computational efficiency is incontestable when the dimension exceeds 20.

Table 5.7: proc.time() in Chapter 4

Time (sec)	Dim	Fu-Wang			Wang-Lee		
		User	System	Total	User	System	Total
Eqn (5.1)	2	81.550	6.270	87.818	59.450	3.320	62.750
Eqn (5.2)	3	21.920	4.040	25.962	20.360	3.560	23.889
Eqn (5.3)	2	61.080	6.770	67.851	46.450	3.970	50.415
Eqn (5.4)	2	738.650	7.200	746.107	810.080	6.320	816.910
Eqn (1.4)	21	19936.72	183.64	20120.72	3409.290	79.080	3489.345

5.3 Practical Guide with Graphical Diagonistic

This section provides the practitioners with a practical guide for implementation of the Wang-Lee algorithm for use in real application. The multimodal example used in Section 5.2.4 is revisited for the purpose of providing the practical standards to identify a verification of appropriateness regarding the drawn sample obtained using the Wang-Lee algorithm.

As mentioned in Section 4.5, the interaction between a sampling procedure and a graphical diagnostic is a critical step in determining the significant region on the discretized sampling space $S_n(f)$. When the significant region is correctly identified

within the compact support $C_0(f)$, the drawn sample is expected to be accepted as the appropriate sample. However, this is not always the case; hence it is necessary to verify the appropriateness of the drawn sample. Figure 5.6 illustrates a failure to obtain a reasonable sample.

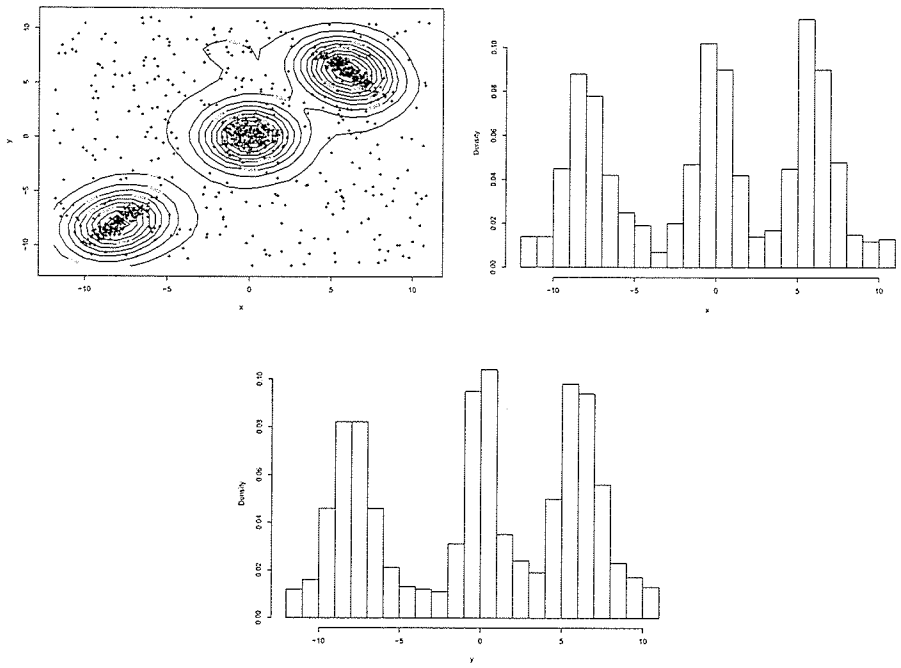


Figure 5.6: Contours, $k = 30$; Weight on Last partition, $w = 0.5$

There are several possible contributing factors that may result in a poor sample. In consideration of programming, it is possible that there is a logical error in the program itself, despite the absence of compile-errors or run-time errors. Such errors are not easily found. It is essential to implement the logically correct program.

Another factor to consider is the situation of an inappropriate sample despite a proper adjustment to the significant region on the sampling space $S_n(f)$ and correct program implementation. A potential resolution for this factor involves the number of partitions. This problem occurred frequently during repeated simulations, when statistical models in the Bayesian paradigm were applied. Mixture model, Hierarchical

model or well-separated multimodal density function all generate similar challenges. This is common to all statistical models when density or posterior distribution have a proportionately long tail with low probability. Caution must be exercised to define a significant region when the statistical model has an appreciable potential for a long tail with low probability.

The implication is that a large proportion of low probability region is found in the last partition; that is, the negligible sampling region is also considered as the significant sampling region. This is an indication that the probability on the last partition is overestimated. This result arises from a characteristic of the Wang-Lee algorithm, which is the horizontal approach for contourizing $f(x)$. Issues concerning this overestimation are addressed and discussed in Section 4.3.2.

This practical guide provides suggestions and techniques to assist with computational and interpretational obstacles with regard to the overestimation of probability on the last partition. Fundamental to the Wang-Lee algorithm is the elimination of the influence from the insignificant region. Visualization is crucial; explicit examination of the tails of distribution on the histogram is vitally important.

To illustrate, if the tails of distribution end within the compact support, it is logical to consider that the sample is appropriate. If the histogram is not fitted accurately, this issue can be resolved by increasing the number of partitions. Since the partitions characterize the target density $f(x)$, the number of partitions enhances the capability of this algorithm to estimate the approximate discrete probability.

Simulation studies in this thesis use a number of 1×10^5 partitions as a default. Increasing the number of partitions provides a reasonably accurate approximation of target density. This neccessitates greater computing resources. Cost efficiency is inversely proportional to computer hardware requirements.

Alternatively, a superior solution to addressing an inappropriate sample arising from an ill-fitted histogram is to weight the last partition without increasing the total

number of partitions. This is pivotal; controlling the weight on the last partition minimizes the redundancy of the last partition. This method is applied to all examples in Chapter 5. The default is set to 0.5 as the Wang-Lee algorithm set the mid point as the level in the last partition for computational convenience. A visual representation of the weight on the last partition is described in Section 4.3.2. Figure 5.7 and 5.8 shows the change on the significant region correponding to the weights.

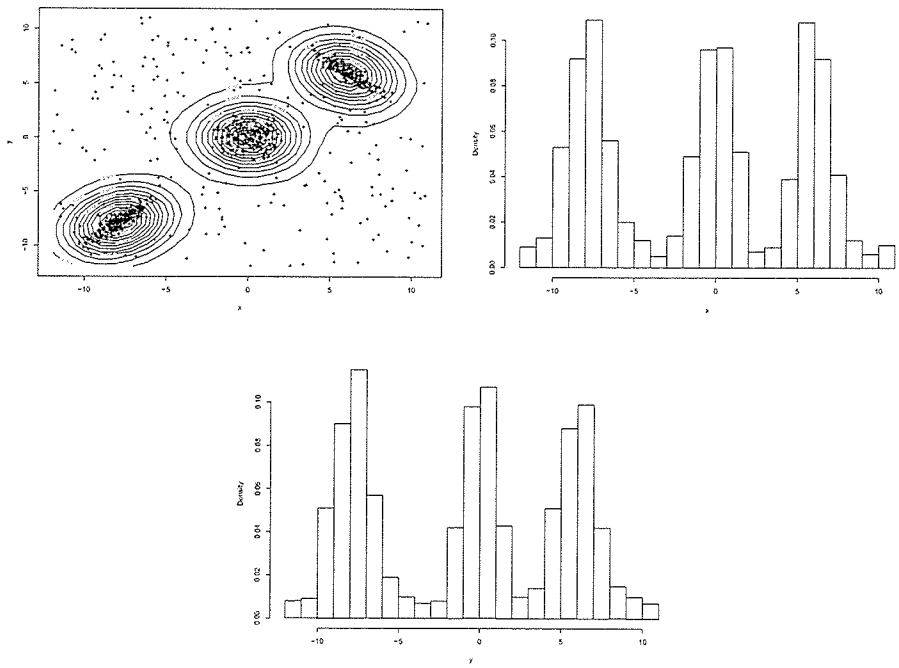


Figure 5.7: Contours, $k = 30$; Weight on Last partition, $w = 0.3$

It is obvious that the samples from the insignificant region are reduced as the weight on last contour decreases.

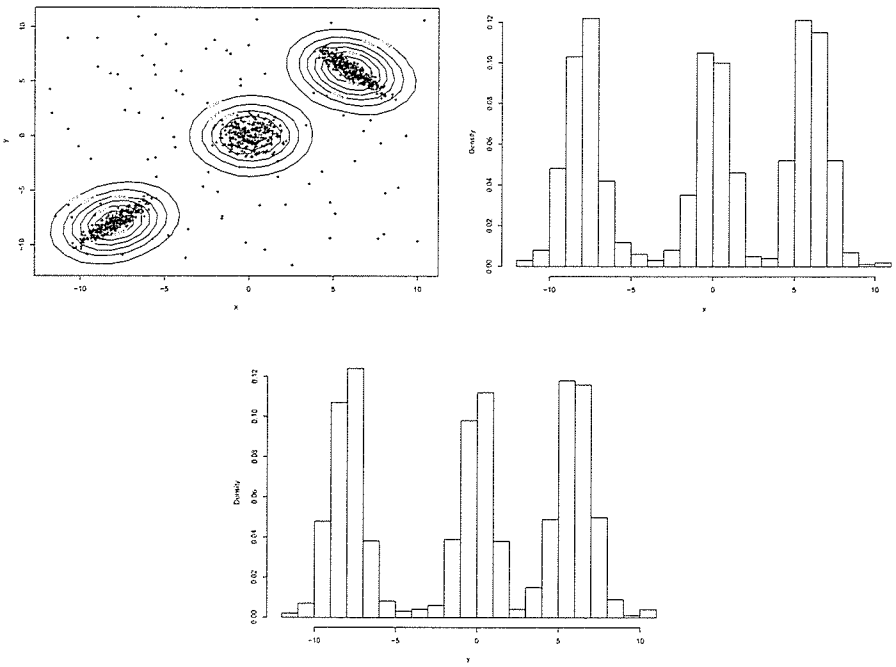


Figure 5.8: Contours, $k = 30$; Weight on Last partition, $w = 0.1$

When the low probability in the tail of the histogram is eliminated, the sample from the significant region becomes appropriate.

Chapter 6

Applications

In Chapter 5, several classic examples were utilized to substantiate the unique characteristics of the Wang-Lee algorithm. To corroborate the efficacy of the Wang-Lee algorithm, applications based on the Bayesian paradigm using a nonlinear model, a hierarchical model, and a mixture model are illustrated here. Chapter 6 develops the functional advantages of the Wang-Lee algorithm — explicitly, freedom from restrictions such as the conjugate priors, the intractable mathematical form, restriction in dimensions, and reparameterization. This is consummated through comparison of computational algorithms for resolution of real applications. Primacy of the Wang-Lee algorithm is incontrovertibly relevant.

6.1 Space Shuttle Challenger Data

Data in Dalal, Fowlkes, and Hoadley's (1989) study was employed to validate the relationship between a probability of *O-ring* failure and the temperature at flight by a logistic model for the resolution of shuttle risk management (pp. 945–952).

This data is utilized to establish the independent Metropolis-Hastings algorithm on the logistic regression model in the study of Robert and Casella (2004) (pp. 281–282).

(For the purpose of comparisons in a simulation study, all notations, the model specification, and prior information in this section are identical as those used in Robert & Casella, 2004, pp. 281–282, the data is located in Table 1.1. p. 15).

$$Y_i \sim \mathcal{B}(p(x_i)), \quad p(x) = \frac{\exp(\alpha + \beta x)}{1 + \exp(\alpha + \beta x)} \tag{6.1}$$

where y_i is the response and $p(x)$ is the probability of an O-ring failure at temperature x. Its prior is

$$\pi_\alpha(\alpha|b)\pi_\beta = \frac{1}{b} e^{\alpha} e^{-e^{\alpha}/b}$$

with $\hat{\alpha}$, MLE from data, $\hat{b} = e^{\hat{\alpha}+\gamma}$, and $\gamma = 0.577216$ (Robert & Casella, 2004, pp. 281–282).

Utilizing only the information provided, the Wang-Lee algorithm is administered to the analytic form of basic full posterior distribution without further reparameterization and adaptation. The full posterior density function has the form, $L(\alpha, \beta|data)\pi(\alpha, \beta)$, then the default simulation values are applied to estimate α and β.

The estimates of α and β are listed in Table 6.1. These estimates are consisitent with others (See MLE from Dalal et al., 1989, p. 949; Metropolis-Hasting(M-H) Algorithm in Robert & Casella, 2004, p. 283).

Table 6.1: α and β of Equation (6.1)

Algorithm	MLE	M-H	Wang-Lee	
Statistics	MEAN	MEAN	MEAN	VARIANCE
α	15.043	15.0	15.1444	1.4284
β	-0.2322	-0.235	-0.2346	0.0003

The histograms of α and β are provided in Figure 6.1 (See also Fig. 7. 3. in Robert

& Casella, 2004, p. 283).

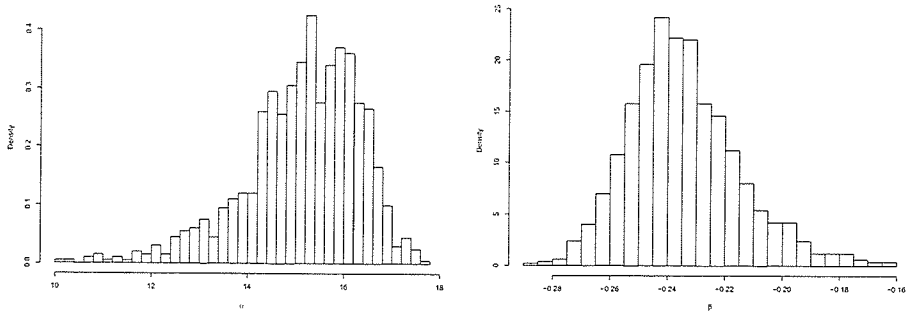


Figure 6.1: Histograms of Equation (6.1)

6.2 Beetle Data

Carlin and Gelfand (1991) employed the observed flour beetle mortality data to demonstrate the Gibbs sampler with a tailor rejection method (pp. 126–127). This data is found in Table 3. (Carlin & Gelfand, 1991, p. 126). The original study was accomplished utilizing the maximum likelihood estimation with the generalized logit model (Prentice, 1976, pp. 761–768). Subsequently, the Metropolis-Hastings algorithm was applied to analyze the same data after the Jacobian transformation on the parameter space in the study of Carlin and Louis (1996) (pp. 176–180).

(For the purpose of comparisons in a simulation study, all notations, the model specification, and prior information in this section are identical to those used in Carlin & Louis, 1996, pp. 176–180).

The generalized logit model as

$$P(\text{death}|w) \equiv h(w) = \{\exp(x)/(1+\exp(x))\}^{m_1}$$

where w is the independent variable and $x = (w-\mu)/\sigma$ with unknown μ and σ^2. Its prior information are $m_1 \sim \mathcal{G}(a_0, b_0)$, $\mu \sim \mathcal{N}(c_0, d_0)$, $\sigma^2 \sim \mathcal{IG}(e_0, f_0)$ with $a_0 = 0.25$,

$b_0 = 4$, $c_0 = 2$, $d_0 = 10$, $e_0 = 2.000004$, and $f_0 = 1000$. The full posterior density is

$$\begin{aligned} p(\mu, \sigma^2, m_1|\mathbf{y}) \propto & f(\mathbf{y}|\mu, \sigma^2, m_1)\pi(\mu, \sigma^2, m_1) \\ \propto & \{\prod_{i=1}^{k}[h(w_i)]^{y_i}[1-h(w_i)]^{n_i-y_i}\}\frac{m_1^{a_0-1}}{\sigma^{2(e_0+1)}} \\ & \times \exp\Big[-\frac{1}{2}(\frac{\mu-c_0}{d_0})^2 - \frac{m_1}{b_0} - \frac{1}{f_0\sigma^2}\Big] \end{aligned} \tag{6.2}$$

To estimate model parameters, the Wang-Lee algorithm generates a sample from the naive full posterior equation 6.2 without any reparameterization. The sampling procedure is terminated when the boundaries of compact supports reache $[1.76, 1.84]$, $[0.0100, 0.0333]$, and $[0.1353, 1.2214]$, for μ, σ and $m1$ respectively. The simplicity and independent sampling is appealing to many practitioners. The Wang-Lee algorithm is reputed as a coherent and convenient alternative to MCMC.

The estimates of μ, σ and $m1$ are listed in Table 6.2. These estimates are consisitent with those that others have found (See MLE from Prentice, 1976, p. 765; Gibb Sampler(G-S) in Carlin & Gelfand, 1991, p. 127; Metropolis-Hastings algorithm(M-H) in Carlin & Louis, 1996, p. 179).

Table 6.2: Means, Standard Deviations, and Modes of μ, σ, m_1 from Equation (6.2)

Statistics	Wang-Lee			Other Algorithms			
	μ	σ	m_1	μ	σ	m_1	
MEAN	1.8163	0.0168	0.3128	1.818	0.016	0.279	MLE
SD	0.0096	0.0030	0.0977	N/A	N/A	N/A	N/A
MODE	1.8172	0.0163	0.2915	1.81	-4.04 (0.0175)	-1.09 (0.3362)	Gibbs
2.5%	1.7951	0.0117	0.1743	1.78	-4.35 (0.013)	-1.61 (0.199)	
50%	1.8170	0.0165	0.2939	1.81	-3.96 (0.019)	-0.98 (0.374)	M-H
97.5%	1.8341	0.0241	0.5706	1.83	-3.6 (0.27)	-0.25 (0.779)	

The histograms of μ, σ and $m1$ are provided in Figure 6.2 (See Fig. 3. in Carlin & Gelfand, 1991, p. 127; Figure 5. 6 in Carlin & Louis, 1996, p. 179)).

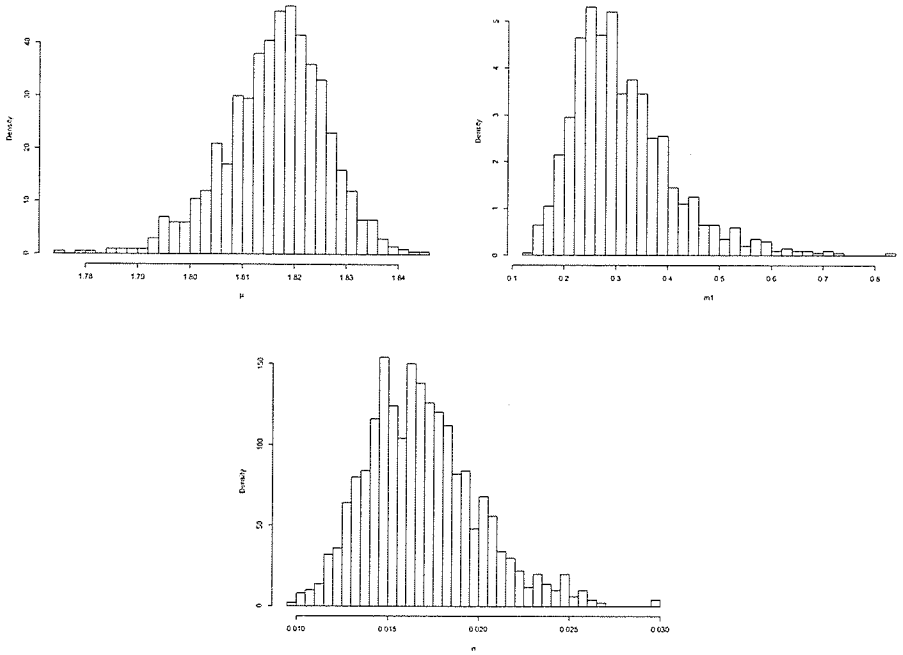


Figure 6.2: Histograms of $\mu, \sigma, m1$ from Equation (6.2)

6.3 Dugong Data

Carlin and Gelfand (1991) employed Sirenian species dugong data to exemplify the failure of Gibbs sampler when the conjugacy does not exist in Bayesian analysis on the growth curve, and applied a tailored rejection method for successful implementation of Gibbs sampler (pp. 124–125, the data is provided in Table 1. p. 124). This data is also used to manifest self-regenerative (SR) algorithm compared with other MCMC algorithms (Sahu & Zhigljavsky, 2003, pp. 412–413). Dugongs data is one of the standard examples in OpenBUGS (For this see Dugongs: nonlinear growth curve Spiegelhalter, Thomas, Best, & Lunn, 2007a).

Recently, Malefaki and Iliopoulos (2008) used the dugongs data to demonstrate the necessity of adaptation on the given density (pp. 1218–1219).

(For consistency of comparison in a simulation study, all notations, model specification, and prior information in this section are identical to those utilized Malefaki & Iliopoulos, 2008, pp. 1218–1219).

The growth curve is

$$y_i \sim \mathcal{N}(\alpha - \beta\gamma^{x_i}, \tau^{-1}) \quad i = 1, \ldots, n \tag{6.3}$$

where α, β, and $\tau > 0$. Its prior information is $\alpha \sim N(0, \tau_\alpha^{-1})I(\alpha > 0)$, $\beta \sim N(0, \tau_\beta^{-1})I(\beta > 0)$, $\gamma \sim U(0,1)$, and $\tau \sim G(k,k)$ with $\tau_\alpha = \tau_\beta = 10^{-4}$ and $k = 10^{-3}$. The likelihood function of parameters α, β, γ, and τ and its naive posterior density function are as follows:

$$L(\alpha, \beta, \gamma, \tau | y_1, y_2, \ldots, y_n) \propto \tau^{\frac{n}{2}+k-1} exp\{-\frac{\tau}{2}\sum_{i=1}^{n}(y_i - \alpha + \beta\gamma^{X_i})^2\} \tag{6.4}$$

$$\begin{aligned} \pi(\theta | y_1, y_2, \ldots, y_n) &\propto f(y_1, y_2, \ldots, y_n | \theta) p(\alpha) p(\beta) p(\gamma) p(\tau) \\ &\propto L(\theta | y_1, y_2, \ldots, y_n) \times exp(-\tau k - \frac{\tau_\alpha \alpha^2}{2} - \frac{\tau_\beta \beta^2}{2}) \\ &\times I(\alpha > 0, \beta > 0, \tau > 0, 0 < \gamma < 1) \end{aligned} \tag{6.5}$$

The Wang-Lee algorithm is straightforwardly applied to the full posterior distribution; further parameterization is not required.

The numerical values in Table 6.3 from the Wang-Lee algorithm are consistent with other algorithms. (See Gibbs Sampler(G-S) from Carlin & Gelfand, 1991, p. 125; OpenBUGS in Dugongs: nonlinear growth curve, Spiegelhalter et al., 2007a). Also, "the least-squares estimates, 0.981, −0.028, and 1.932, obtained by Ratkowsky (1983, p. 96)" (Carlin & Gelfand, 1991, p. 125).

Table 6.3: Means and Modes of α, β, γ of Equation (6.5)

		Wang-Lee	OPENBUGS	LSE
MEAN	α	2.6601 (0.9783)	2.652 (0.9753)	0.981
	β	0.9791 (-0.0210)	0.9729 (-0.0274)	-0.028
	γ	0.8642 (1.8509)	0.8623 (1.8345)	1.932
		Wang-Lee	G-S	
MODE	α	2.6710 (0.9824)	0.975	
	β	0.9668 (-0.0337)	-0.014	
	γ	0.8770 (1.9646)	1.902	

The histograms of α, β, and γ are provided in Figure 6.3 (See Fig. 1. in Carlin & Gelfand, 1991, p. 124; Fig. 4. (a) and (b) in Malefaki & Iliopoulos, 2008, p. 1219).

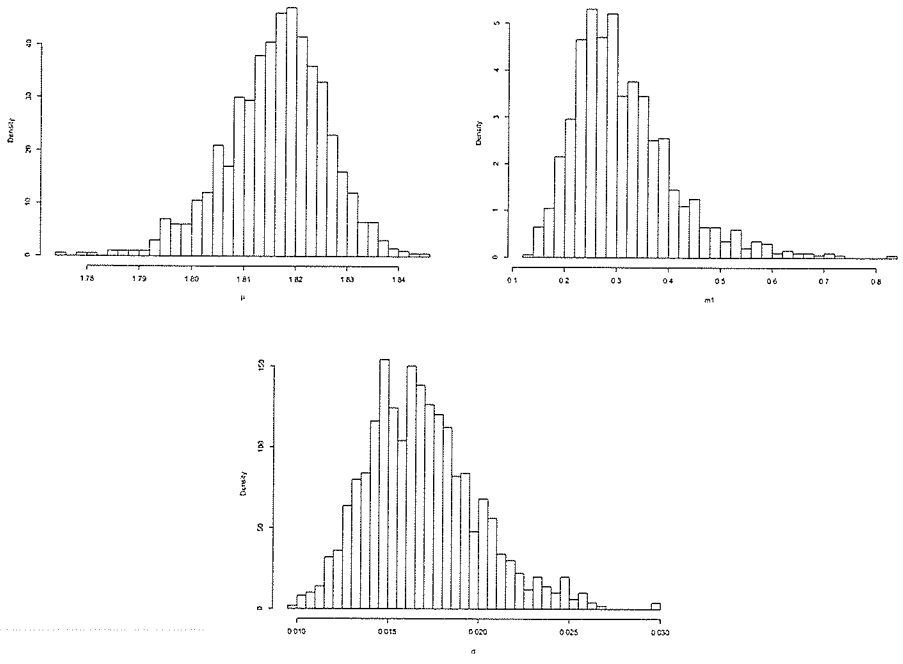


Figure 6.3: Histograms of $\mu, \sigma, m1$ from Equation (6.5)

6.4 British Coal Mining Data

Previously in this thesis, the Wang-Lee algorithm has been implemented to only those cases of low dimensions (less than four parameters). The following three applications will demonstrate that the Wang-Lee algorithm provides results consistent with other statistical algorithms notwithstanding high dimensions. This section illustrates a case of five dimensions using the data from British coal-mining disasters, 1851-1962 (Tanner, 1996, pp. 147–149, and the data is found in Table 6. 1. p. 148).

Carlin, Gelfand, and Smith (1992) carried out research for obtaining the modes of marginal posterior distributions of a three-stage hierarchical model by Gibbs sampler (pp. 393–400). The features of the Fu-Wang algorithm were demonstrated utilizing this example (for the purpose of comparisons in a simulation study, all notations, model specification, and prior information in this section are identical to those used in Fu & Wang, 2002, pp.18–20).

Fu and Wang (2002) postulated the model specification as follows: $X_i \sim \text{Poi}(\theta t_i)$, $i = 1, 2, \ldots, \kappa$, $X_i \sim \text{Poi}(\lambda t_i)$, $t = \kappa + 1, \ldots, N$ at the first stage. They expressed the log-likelihood function analytically as equation (6.6) to find the approximate mode.

$$\mathcal{L}(\kappa, \theta, \lambda) = \left(\sum_{i=1}^{\kappa} x_i - 1/2 \right) \log \theta + \left(\sum_{i=\kappa+1}^{n} x_i - 1/2 \right) \log \lambda - \kappa\theta - (n - \kappa)\lambda, \quad (6.6)$$

where $\kappa \in (1 : N)$, $\theta \in (0, \infty)$, and $\lambda \in (0, \infty)$. The prior information is $\kappa \sim \mathcal{U}(1, N)$, $\theta \sim \mathcal{G}(1/2, \alpha)$, and $\lambda \sim \mathcal{G}(1/2, \beta)$ at the second stage and $\alpha \sim \mathcal{IG}(2, 1)$ and $\beta \sim \mathcal{IG}(2, 1)$ at the third stage. The full log-posterior distribution is

$$f(\kappa, \theta, \lambda, \alpha, \beta) = l(\kappa, \theta, \lambda) + 1.5 \log \alpha + 1.5 \log \beta - (\theta + 1)\alpha - (\lambda + 1)\beta. \quad (6.7)$$

The boundaries of compact support are applied as follows: $\kappa \in (30 : 50)$, $\theta \in [2.2, 4]$, $\lambda \in [0.6, 1.4]$, $\alpha \in [0, 2]$ and $\beta \in [0, 4]$ (Fu & Wang, 2002, pp. 17–18).

The mode information is contained in Table 6.4. See Gibb Sampler in Carlin et al. (1992, p. 397).

Table 6.4: Approximated MLE, Means, Standard Deviations, and Modes of κ, θ, λ, α, and β from Equation (6.7)

	Wang-Lee				
	κ	θ	λ	α	β
AMLE	41	3.0858	0.8945	N/A	N/A
MEAN	40.0160	3.0765	0.9118	0.6250	1.2815
SD	2.4778	0.2826	0.1133	0.3667	0.7608
MODE	41	2.9957	0.8995	0.3905	0.7629
	Gibbs Sampler				
	κ	θ	λ	α	β
MODE	41	3.06	0.89	N/A	N/A

The marginal posterior distributions are shown as Figure 6.4 (See also Figure 7. in Fu & Wang, 2002, p. 21; Fig. 1. in Carlin et al., 1992, p. 395).

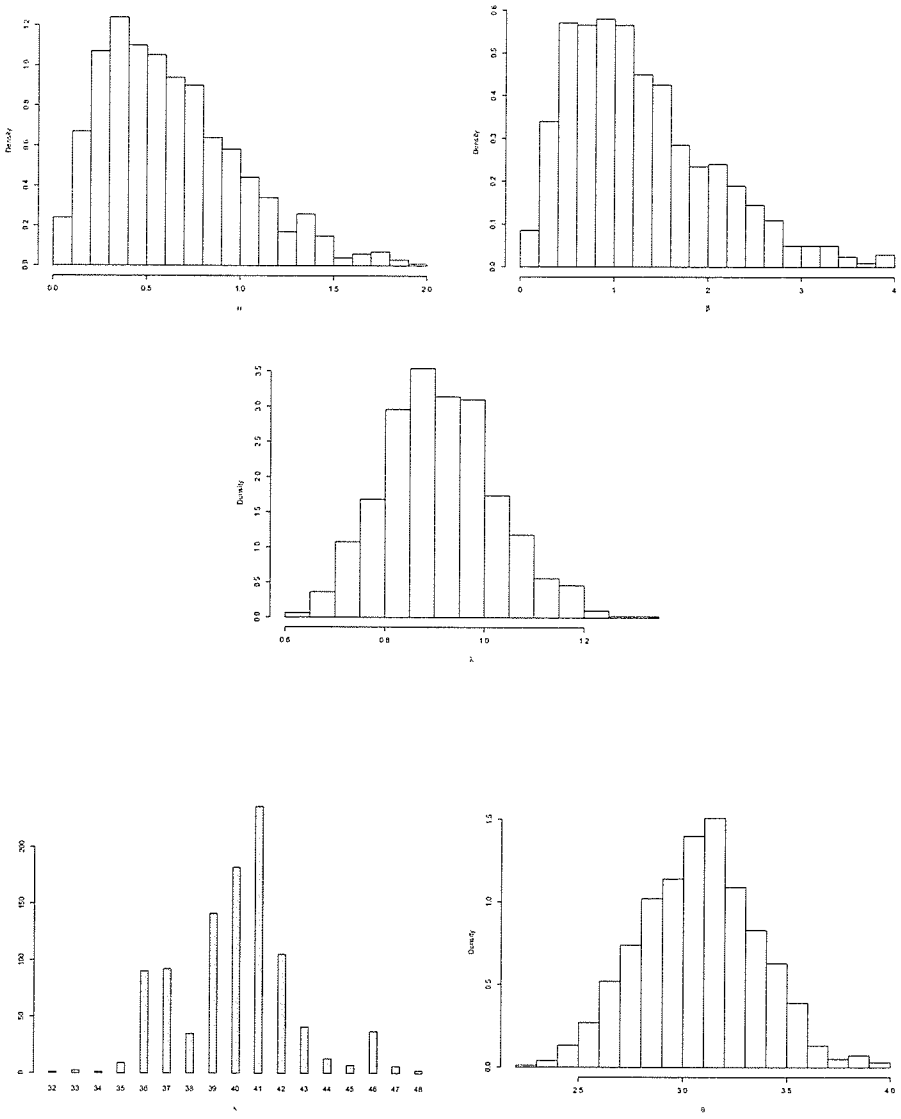


Figure 6.4: Histograms of α, β, λ, κ, and θ from Equation (6.7)

6.5 Nuclear Pump Data

Previous application of five dimensions has established that the Wang-Lee algorithm is capable of presenting estimates consistent with other MCMC algorithms in low and moderate dimensions. In the next application, the validity of the Wang-Lee

algorithm will be affirmed by challenging a real application consisting of ten dimensions.

Gaver and O'Muircheartaigh (1987) employed the failure of pump data to explain the failure rate of individual pumps for reliability management by employing the parametric empirical Bayesian model (pp. 1–14). It is also utilized for the illustration of Gibb Sampler in the hierarchical model in the study of Robert and Casella (2004) (pp. 385–387).

(For the purpose of comparisons in a simulation study, all notations, model specification, and prior information in this section are identical to those used in Robert & Casella, 2004, pp. 385–387, corresponding data is found in Table 10. 1. p. 386.)

The model is described as follows: the number of failures $p_i \sim \text{Poi}(\lambda_i t_i)$ with $i \leq i \leq 10$ and its priors are $\lambda_i \sim \mathcal{G}(\alpha, \beta)$ and $\beta \sim \mathcal{G}(\gamma, \delta)$ with $\alpha = 1.8$, $\gamma = 0.01$, and $\delta = 1$. The full posterior has the form as

$$\begin{aligned} &\pi(\lambda_1, \ldots, \lambda_{10}\beta | t_1, \ldots, t_{10}, p_1, \ldots, p_{10}) \\ &\propto \prod_{i=1}^{10} \{(\lambda_i t_i)^{p_i} e^{-(t_i+\beta)\lambda_i}\} \beta^{10\alpha+\gamma-1} e^{-\delta\beta} \end{aligned} \tag{6.8}$$

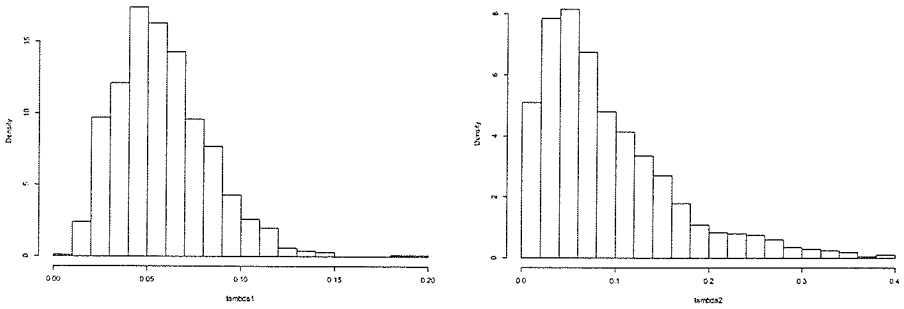
This simulation study is slightly altered by adopting different priors in advance of launching the Wang-Lee algorithm procedure. The priors are taken as $\alpha = 0.54$, $\gamma = 2.20$, and $\delta = 1.11$ because the prior information is subjective on the basis of a researcher's conviction regarding the problem if a true Bayesian approach is considered (Dagpunar, 2007, pp. 169–170).

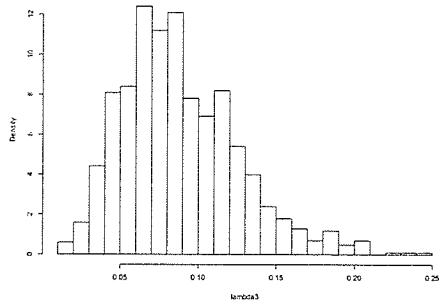
The numerical values in Table 6.5 from the Wang-Lee algorithm are consistent with other algorithms (See Bayes in Dagpunar, 2007, p. 168; OpenBUGS in Pumps: conjugate gamma-Poisson hierarchical model Spiegelhalter et al., 2007b).

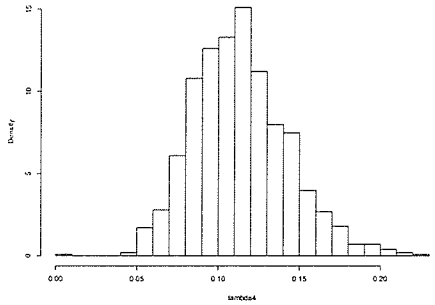
Table 6.5: Rates, Modes, Means, and Standard Deviations of Pumps λ_i, $i = 1, 2, \ldots, 10$ from Equation (6.8)

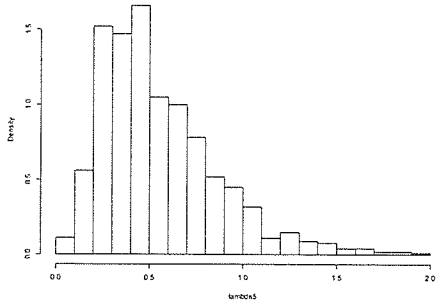
		Wang-Lee			Dagpunar	OPENBUS
PUMP	RATE	MODE	MEAN	SD	Bayes	MEAN
λ_1	0.0530	0.0586	0.0581	0.0249	0.0581	0.0598
λ_2	0.0636	0.0645	0.0903	0.0708	0.0920	0.1015
λ_3	0.0795	0.0775	0.0883	0.0376	0.0867	0.0889
λ_4	0.1113	0.1147	0.1135	0.0291	0.114	0.1156
λ_5	0.5725	0.4185	0.5509	0.3121	0.566	0.6043
λ_6	0.6043	0.5318	0.6061	0.1343	0.602	0.6121
λ_7	0.9523	0.2370	0.6705	0.4739	0.764	0.899
λ_8	0.9523	0.1199	0.6810	0.4737	0.764	0.9095
λ_9	1.9047	2.0381	1.4448	0.6743	1.470	1.587
λ_{10}	2.0992	2.0105	1.9467	0.4044	1.958	1.995

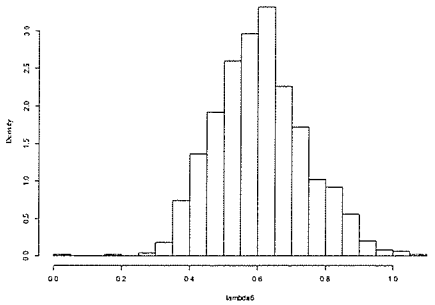
Histograms are listed in Figure 6.5 (See also Figure 8. 2. Posterior generated by Gibbs sampling by MAPLE in Dagpunar, 2007, p. 170).

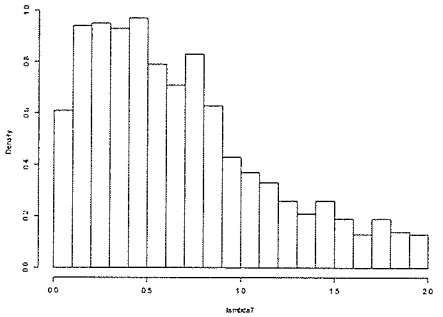


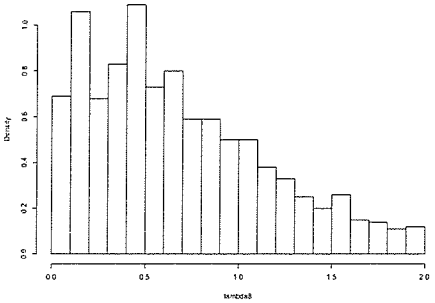

Density
lambda3


Density
lambda4


Density
lambda5


Density
lambda6


Density
lambda7


Density
lambda8

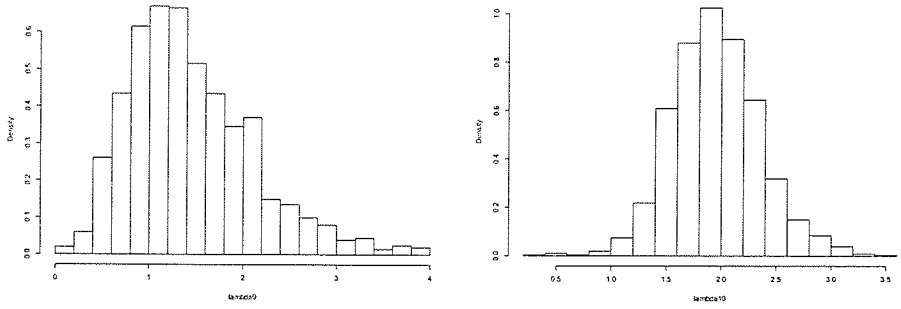


Figure 6.5: Histograms of λ_i, $i = 1, 2, \ldots, 10$ from Equation (6.8)

6.6 SLC 190 Genetic Data

The dimension-free characteristic of the Wang-Lee algorithm has been comprehensively illustrated in previous applications. Significant emphasis has been placed on the high computational efficiency of the Wang-Lee algorithm when applied to instances wherein the number of dimensions is greater than or equal to 10. For this endeavor, the number of parameters to be estimated is 21. The computing time for all applications in Chapter 6 are reported in Table 6.8.

This application is a follow-up study of the simulated data analysis in Section 5.2.5. It originated from hypotheses testing of two statistical models, the decision regarding the number of genotypes, in Roeder's 1994 study. He analyzed the SLC190 data by the utilization of graphical diagnostics to test and identify the total components in a mixture model regarding difficulties arising from the generalized likelihood ratio test and model selection (pp. 487–495, data is found in Table 1. 492). For this data, the Fu-Wang algorithm is applied with the equivalent model and priors specification on a Bayesian hierarchical mixture model with unknown sub-populations (to faciliate comparisons in a simulation study, all notations, model specification, and prior information in this section are identical to those used in Wang & Fu, 2007, pp. 645–647).

Implementing the Wang-Lee algorithm, the statistical model and priors information are identical to those used in Section 5.2.5. The likelihood function and the full posterior distributions are identical to the equation (1.2) and (1.4) respectively. The different conditions are the specification of the compact support such that $\mu^K \in [0,7]^K$, $\sigma_K^2 \in [0.1,1]^K$ and $w^K \in [0,1]^K$ and the prior $K = (0.2, 0.3, 0.3, 0.2)$.

Table 6.6 and Table 6.7 include the estimated marginal posterior distribution concerning the number of component K and approximated posterior modes, means and standard deviations are provided respectively (See MLE in Roeder, 1994, p. 492)

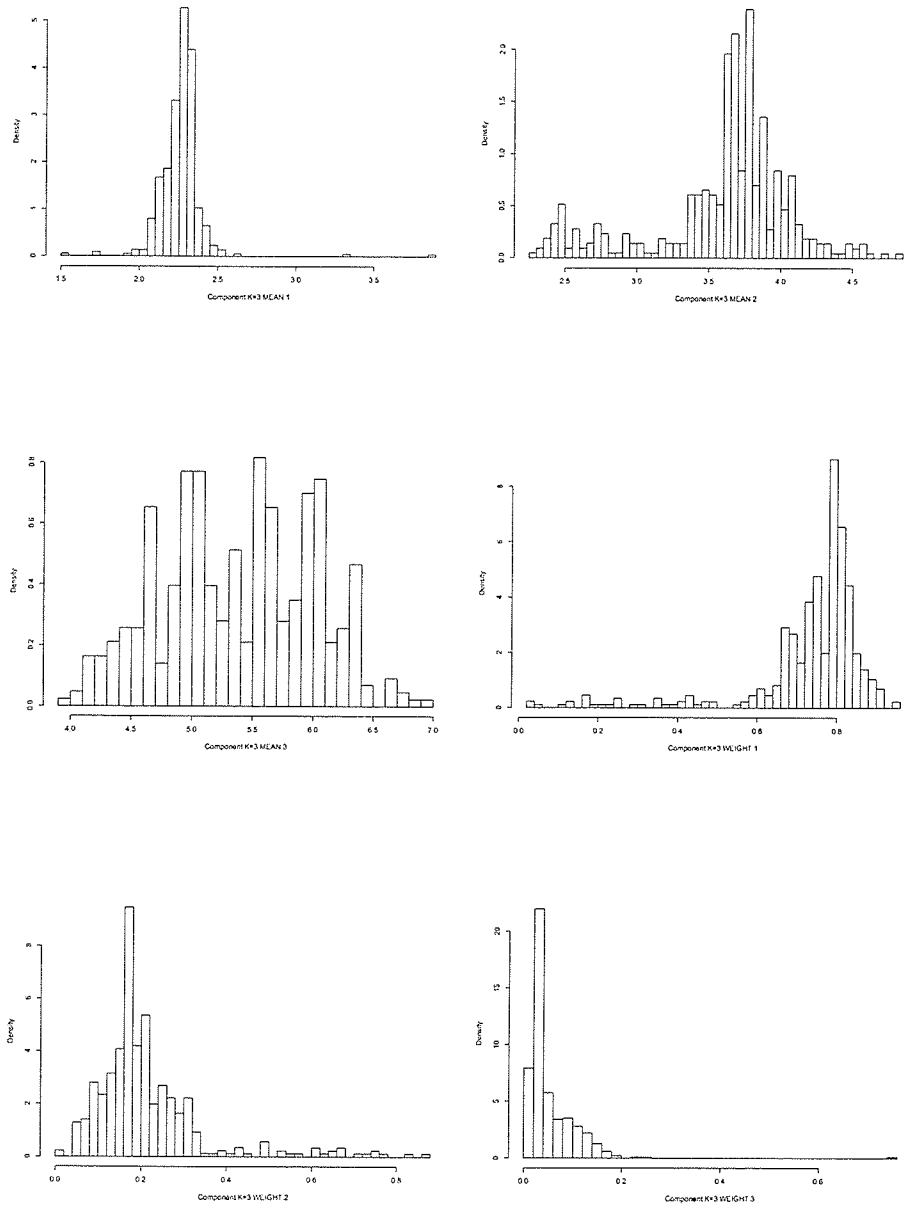
Table 6.6: Prior and Posterior distribution of K from Equation (1.4)

K	1	2	3	4
Prior	0.2	0.3	0.3	0.2
Fu-Wang	0	0.238	0.404	0.358
Wang-Lee	0.005	0.311	0.408	0.276

Table 6.7: Parameters, Posterior Means and Standard Deviations of μ^3 and w^3 for $K = 3$ from the Equation (1.4)

$K = 3$	Fu-Wang		Wang-Lee	
MLE	MEAN	SD	MEAN	SD
$\mu_1^3 = 2.23$	2.2443	0.1441	2.2503	0.2335
$\mu_2^3 = 3.79$	3.5760	0.4791	3.6173	0.4878
$\mu_3^3 = 5.77$	5.3852	0.5785	5.4520	0.6664
$w_1^3 = 0.774$	0.7261	0.1699	0.7304	0.1663
$w_2^3 = 0.202$	0.2239	0.1536	0.2159	0.1466
$w_3^3 = 0.024$	0.0498	0.0409	0.0535	0.0648
σ^3	0.4071	0.0861	0.4163	0.0914

Histograms of the marginal distributions are listed in Figure 6.6 (See also Figure 9 and Figure 10 in Wang & Fu, 2007, p. 648).



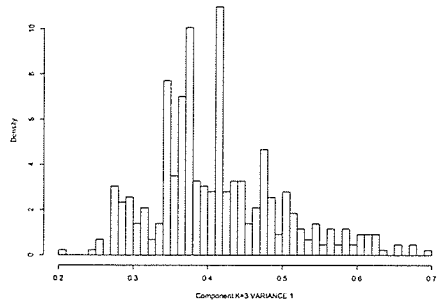


Figure 6.6: Histograms of $\mu_i^{K=3}$, $\sigma^{K=3}$, and $w_i^{K=3}$ from Equation (1.4)

Comparison of the computing time is provided in Table 6.8, Section 6.7. In this application it is remarkable that the Wang-Lee algorithm is established at approximately eight times faster in 21 dimensions of Bayesian hierarchical model when compared to the Fu-Wang algorithm.

6.7 Computing Times

In real time applications, there is little difference in computing time for cases of less than 5 dimensions between the Fu-Wang algorithm and the Wang-Lee algorithm. In dimensions greater than 5, significant computational efficiency is found to be evident. All running times are listed on Table 6.8.

Table 6.8: proc.time() in Chapter 6

Time (sec)	Dim	Fu-Wang			Wang-Lee		
		User	System	Total	User	System	Total
Eqn (6.1)	2	182.310	35.790	218.099	181.390	34.370	215.752
Eqn (6.2)	3	130.570	24.110	161.694	113.55	20.640	134.19
Eqn (6.5)	3	122.570	23.290	145.854	92.250	20.220	112.473
Eqn (6.7)	5	69.370	13.090	82.445	60.580	9.770	70.348
Eqn (6.8)	10	102.940	25.240	128.202	81.850	18.010	99.874
Eqn (1.4)	21	25871.86	190.28	26063.04	3577.530	17.660	3595.237

Chapter 7

Summary and Further Research

This thesis improves the discretization-based Monte Carlo algorithm of Fu and Wang (2002) for a random variate generation from a complicated form of high dimensional distributions. The essence of the Wang-Lee algorithm is the approximation of the discrete probability function of partitions derived from the discretized sample space and the discrete inversion of multivariate cumulative density of partitions by a two-stage sampling scheme.

From the perspective of computational practice, the Wang-Lee algorithm contributes to the computational efficiency of the Fu-Wang algorithm by the shift of direction on contourization. This efficiency is amplified by the proposition of adjustment on the probability of the last partition. This is illustrated by examples and applications in Chapters 5 and 6. The dimension-free, non-iterative procedures and the ease of implementation of this algorithm enhances the scope of functional applications for computational statisticians. Provision of the generalized R program with the practical guide enables immediate utilization of this algorithm by other researchers.

In consideration of statitstical inference, a fundamental asset of the Wang-Lee algorithm is the inherent capability of detecting multimodes in a mixture model. Additional advantages include abundant statistics of interest and the analytical func-

tional form without the parameterization or other proper adaptation. Particularly, in Bayesian framework the normalizing constant is not necessary. The independent random variates by the Wang-Lee algorithm enable the construction of a confidence interval as well. These assets compensate for the recognized limitations ascribed to the standard MCMC algorithms.

To be circumspect when applying the Wang-Lee algorithm, a particular caution is urged in the utilization of graphical diagnostics for a validation of the relevant sample from the significant region and establishment of the correct sampling space. Careful attention is required in eliminating the consequences of overestimation on the region of a large proportion of low tail probability. This facilitates the more accurate construction of approximate discrete probability function and contribution to the high performance computation.

A number of questions remain for future development of the Wang-Lee algorithm. Further research is required in the following areas:

1. **The optimal number of partitions** k: Since the approximate discrete probability function characterizes the configuration of distribution based on the number of partitions, it is considered to distinguish the optimal number of partitions to quantify the discrete probability function regardless of the number of dimensions. In example 5.2.5, application 6.5, and application 6.6, the optimal histograms of marginal distributions and samples are found when $k = 10^6$ higher up to $k = n$. The implication is that the number of partitions is insufficient to precisely characterize the approximate discrete probability function in high dimensions (greater than ten). As previously stated, a greater incidence of partitions generates a more concise approximation though the loss of computational efficiency. A designation of the appropriate weight on the last partition is the provisional trial for the resolution of this concern.

2. **Using the parallel algorithm**: Currently, computational technology is pro-

gressing with a multicore system or a high-performance distributed computing system. The performance of parallel computing and its impacts are clearly shown (See Figure 2 Vera, Jansen, & Suppi, 2008, p. 390). The optimal parallel algorithm design is important, a major source to the utilization of these systems depending on the number of cores or distributed nodes. A study of parallel implementation of this algorithm should be considered for further research.

3. **The integration with the numerical analysis on the last partition**: The probability on the last partition is weighted with the arbitrary constant, w, after obtaining the approximate discrete probability function. One of the suggestions for the determination of this weight is the regression analysis between the indice of sorted discrete base points $x^{[j]}$ and $f(x^{[j]})$ in two dimensions. The coefficient of intercept in this regression line indicates the proportion to the height of the last partition, and the coefficient of slope is always negative because $f(x^{[j]})$ is sorted in descending order. The weight is determined in the proportion of the area lower than the regression line to the size of small rectangle of the partition. The other suggestion is to utilize the numerical analysis such as the Rieman sums, trapzoidal, or Simpson's rules in the low probability region (Robert, 2007, p. 293). A study of finding closer and more accurate discrete probability function is required for the true density function.

4. **Quasi-Monte Carlo method**: Fishman (2006) briefly introduces the alternative of Monte Carlo methods to improve both the numerical accuracy and the computational efficiency in high dimensions — the utilization of quasirandom numbers (p. 182). The matter of dimensionality and its inverse relationship with the computational efficiency are the central problem in Monte Carlo method. This matter is also a consideration with the Wang-Lee algorithm despite computational efficiency in high dimensions. The incorporation of QMC on the

Wang-Lee sampling algorithm merits further research.

References

Carlin, B. P., & Gelfand, A. E. (1991). An iterative Monte Carlo method for non-conjugate Bayesian analysis. *Statistics and Computing*, *1*(2), 119–128.

Carlin, B. P., Gelfand, A. E., & Smith, A. F. M. (1992). Hierarchical Bayesian analysis of changepoint problems. *Applied Statistics*, *41*(2), 389–405.

Carlin, B. P., & Louis, T. A. (1996). *Bayes and Empirical Bayes Methods for Data Analysis.* New York : Chapman & Hall.

Celeux, G., Hurn, M., & Robert, C. P. (2000). Computational and inferential difficulties with mixture posterior distributions. *Journal of the American Statistical Association*, *95*(451), 957–970.

Dagpunar, J. S. (2007). *Simulation and Monte Carlo: With applications in finance and MCMC.* Hoboken, NJ : John Wiley.

Dalal, S. R., Fowlkes, E. B., & Hoadley, B. (1989). Risk analysis of the space shuttle: Pre-challenger prediction of failure. *Journal of the American Statistical Association*, *84*(408), 945–957.

Devroye, L. (1986). *Non-Uniform Random Variate Generation.* New York : Springer-Verlag.

Fishman, G. S. (2006). *A First Course in Monte Carlo.* Toronto: Thomson Brooks/Cole.

Fu, J., & Wang, L. (2002). A random-discretization based Monte Carlo sampling method and its applications. *Methodology And Computing In Applied Probabil-*

ity, *4*, 5–25.

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

Gentle, J. E. (2002). *Elements of Computational Statistics: with 86 illustrations.* Retrieved from `http://www.springerlink.com.proxy1.lib.umanitoba.ca/content/g01223/?p=cb8122acbb4a497e8f20f9f564acc013&pi=0`.

Gentle, J. E. (2003). *Random Number Generation and Monte Carlo Methods* (2 ed.). New York : Springer.

Gilks, W., Richardson, S., & Spiegelhalter, D. (1998). *Markov Chain Monte Carlo in Practice: Interdisciplinary Statistics.* Boca Raton, Fla: Chapman & Hall/CPC.

Gilks, W. R., Roberts, G. O., & Sahu, S. K. (1998). Adaptive Markov chain Monte Carlo through regeneration. *Journal of the American Statistical Association*, *93*(443), 1045–1054.

Liang, F., Liu, C., & Carroll, R. J. (2007). Stochastic approximation in Monte Carlo computation. *Journal of the American Statistical Association*, *102*(477), 305–320.

Liu, J. S., Liang, F., & Wong, W. H. (2001). A theory for dynamic weighting in Monte Carlo computation. *Journal of the American Statistical Association*, *96*(454), 561–573.

Lohr, S. L. (1999). *Sampling: Design and analysis.* Scarborough, ON: Duxbury Press.

Madras, N. N. (2002). *Lectures on Monte Carlo Methods.* Providence, R.I. : American Mathematical Society.

Malefaki, S., & Iliopoulos, G. (2008). On convergence of properly weighted samples to the target distribution. *Journal of Statistical Planning and Inference*, *138*(4), 1210 – 1225.

Oehlert, G. W. (1998). Faster adaptive importance sampling in low dimensions.

Journal of Computational and Graphical Statistics, *7*(2), 158–174.

Prentice, R. L. (1976). A generalization of the probit and logit methods for dose response curves. *Biometrics*, *32*(4), 761–768.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical Recipes: The Art of Scientific Computing* (3 ed.). New York : Cambridge University Press.

R Development Core Team. (2008a). *Memory Limits in R.* posted on `http://stat.ethz.ch/R-manual/R-patched/library/base/html/Memory-limits.html` Package base version 2.9.1.

R Development Core Team. (2008b). *Numerical Characteristics of the Machine.* posted on `http://stat.ethz.ch/R-manual/R-patched/library/base/html/zMachine.html` Package base version 2.9.1.

R Development Core Team. (2008c). *R: A Language and Environment for Statistical Computing.* Vienna, Austria. (ISBN 3-900051-07-0)

R Development Core Team. (2008d). *Running Time of R.* posted on `http://stat.ethz.ch/R-manual/R-patched/library/base/html/proc.time.html` Package base version 2.9.1.

Robert, C. P. (2007). *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation* (2 ed.). New York : Springer.

Robert, C. P., & Casella, G. (2004). *Monte Carlo Statistical Methods* (2 ed.). New York : Springer.

Roeder, K. (1994). A graphical technique for determining the number of components in a mixture of normals. *Journal of the American Statistical Association*, *89*(426), 487–495.

Ross, S. M. (1996). *Simulation.* San Diego: Harcourt / Academic Press.

Sahu, S. K., & Zhigljavsky, A. A. (2003). Self-regenerative Markov chain Monte Carlo with adaptation. *Bernoulli*, *9*(3), 395–422.

Spiegelhalter, D., Thomas, A., Best, N., & Lunn, D. (2007a). *Dugongs: non-linear growth curve.* posted on `http://mathstat.helsinki.fi/openbugs/Examples/Dugongs.html`.

Spiegelhalter, D., Thomas, A., Best, N., & Lunn, D. (2007b). *Pumps: conjugate gamma-Poisson hierarchical model.* posted on `http://mathstat.helsinki.fi/openbugs/Examples/Pumps.html`.

Tanner, M. A. (1996). *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions* (3 ed.). New York : Springer-Verlag.

Vera, G., Jansen, R., & Suppi, R. (2008). R/parallel - speeding up bioinformatics analysis with R. *BMC Bioinformatics*, *9*(1), 390.

Wang, L., & Fu, J. (2007). A practical sampling approach for a Bayesian mixture model with unknown number of components. *Statistical Papers*, *48*(4), 631–653.

Wang, L., & Lee, C. H. (2008). *Efficient Monte Carlo random sample generation through discretization : Directional shift on contourization.*

West, M. (1993). Approximating posterior distributions by mixtures. *Journal of the Royal Statistical Society. Series B (Methodological)*, *55*(2), 409–422.

Appendix A

R Programs

Three sample programs are provided for the case of bounded support, the case of unbounded support, and application for a simulated data. R programs for other examples and applications used in this thesis are available upon request.

A.1 Bivariate Beta Distribution

```
1 # Filename : beta.wl.R [Final Thesis Submission Program]
2 # Date: 2009.07.15
3 # Programmer : Chel Hee Lee
4 # Example : Beta Distribution
5 # Paper : Random Discretization Based on Monte Carlo, Fu & Wang 2002
6 # Type : Wang-Lee Algorithm, Bounded Case
7
8 rm(list=ls(all.names=TRUE));
9
10 # Wang-Lee Algorithm Definition
11 wanglee <- function(n.discrete.pnts=1e7, n.cnts=1e5, n=1e3, last.weight
      =0.5, n.analysis="TRUE", g.hist="TRUE", g.plot="TRUE", seed
      =584479233, ... ){
12
13   cat(" *****   Program Description   ***** \n");
14   cat("Filename : beta.wl.R \n");
15   cat("Description : Beta Distribution with Wang-Lee Algorithm\n");
16   cat("Affiliation : Statistics, University of Manitoba \n");
17   cat("Supervisor  : Dr. Liqun Wang, wangl1@cc.umanitoba.ca \n");
18   cat("Programmer  : Chel Hee Lee, umlee@cc.umanitoba.ca, gnustats@gmail
       .com \n\n");
19
20   cat(paste("Program Launching Time : ", Sys.time(),"\n\n" ));
21
22   # Random Number Specification
23   set.seed(seed);
24   cat(paste("Random Seed = ", seed, "\n\n"));
25
26   # Given parameters
```

```
27  alpha <- c(2, 3);
28  beta <- c(2, 1);
29
30  # Forming a discretized base
31  x1 <- runif(n.discrete.pnts, 0, 1);
32  x2 <- runif(n.discrete.pnts, 0, 1);
33
34  cat("*****  Simulation Configuration  *****\n");
35  cat(paste("Number of Discrete Base Points = ", n.discrete.pnts, "\n"))
      ;
36  cat(paste("Number of Partitions = ", n.cnts, "\n"));
37  cat(paste("Weight on Last Partition = ", last.weight, "\n"));
38  cat(paste("A Size of Sample drawn = ", n, "\n\n"));
39
40  cat("Given Values on Parameters \n");
41  for(ind in 1:2){
42    cat(paste("alpha", ind, " = ", alpha[ind], " beta", ind, " = ", beta
        [ind], "\n"));
43  }
44  cat("\n\n");
45
46  # Defining the target distribution
47  density <- dbeta(x1, alpha[1], beta[1])*dbeta(x2, alpha[2], beta[2]);
48
49  # Compact Region
50  monte.space <- data.frame( x1=x1, x2=x2, density=density );
51  rm(list=c("x1", "x2"));
52
53  cat(paste("Discretization procedure is completed at ", Sys.time(),"\n\
      n" ));
54
55  cat("***** Trimming Information ***** \n");
56  density <- density[ which(density > 0) ];
57  cat(paste("Number of trimmed zeros in the density = ", n.discrete.pnts
      -length(density), "\n"));
58  cat(paste("Number of remained discrete base points = ", length(density
      ), " Remaining Rate = (", 1 - (n.discrete.pnts-length(density))/
      length(density),")\n\n"));
59
60  cat("***** Quantiles of Density After Trimming ******\n");
61  print(summary(density));
62  chk.quantile <- seq(0.1, 1, 0.1);
63  print(quantile(density, chk.quantile));
64  cat("\n\n");
65
66  # Sorting by decending order
67  density.ind <- order(density, decreasing=TRUE);
68
69  # Contourization
70  lebergue.measure <- hist(density, breaks=seq(from=min(density), to=max
      (density), length.out=n.cnts+1), plot=FALSE);
71
72  # CDF
73  get.pdf <- rev(lebergue.measure$counts * lebergue.measure$mids);
```

```
74  get.pdf[n.cnts] <- last.weight * get.pdf[n.cnts];
75  nmlzd.cdf <- c(0, cumsum(get.pdf)/sum(get.pdf));
76
77  # Sampling
78  rnd.variates = runif(n);
79  p.sample.size <- hist(rnd.variates, breaks=nmlzd.cdf, plot=FALSE)$
        counts;
80  valid.samples <- which(p.sample.size>0);
81
82  cat("*****   Analysis of samples   ***** \n");
83  cat(paste("Number of Partitions used for sampling = ", length(valid.
        samples), "\n\n"));
84
85  # Searching indices from the sample space
86  sample.cnt <- mapply(sample, MoreArgs=list(replace=TRUE), rev(lebergue
        .measure$counts), p.sample.size);
87  cum.pnts.ind <- c(0, cumsum(rev(lebergue.measure$counts)));
88  cum.pnts.ind <- cum.pnts.ind[-(n.cnts+1)];
89  sample.list <- mapply("+", as.list(cum.pnts.ind), sample.cnt);
90  sample.ind <- unlist(sample.list);
91  rm(list=c("lebergue.measure", "p.sample.size", "rnd.variates", "sample
        .cnt", "cum.pnts.ind", "sample.list"));
92
93  # Saving the final samples
94  monte.space <- monte.space[density.ind[sample.ind],];
95
96  cat(paste("Sampling procedure is completed at ", Sys.time(),"\n\n" ));
97
98  # Numerical Analysis
99  numeric.analysis <- function(){
100   cat(" *****   Numerical Aanlysis   ***** \n");
101   meanMarginal <- sapply(monte.space, mean);
102   sdMarginal <- sqrt( sapply(monte.space, var) );
103   return(list(MEAN=meanMarginal, SD=sdMarginal));
104 }
105
106 # Graphical Analysis (Histogram)
107 graphic.hist <- function(){
108   parameters <- names(monte.space);
109   true.beta.shape1 <- alpha;
110   true.beta.shape2 <- beta;
111
112   for(p.ind in 1:(length(parameters)-1)){
113     graphic.cmd <- gsub(" ", "", paste("postscript(file='beta.wl.x", p
          .ind, ".eps')"));
114     eval(parse(text=graphic.cmd));
115     hist.cmd <- gsub(" ", "", paste("with(monte.space, hist(x", p.ind,
          ", prob=TRUE, freq=FALSE, xlim=c(0,1), breaks=30, xlab=
          expression(", p.ind,"), main=''));"));
116     eval(parse(text=hist.cmd));
117
118     lines(density(monte.space[,p.ind]), lty=1);
119     x <- rbeta(nrow(monte.space), true.beta.shape1[p.ind], true.beta.
          shape2[p.ind]);
```

```
120       curve(dbeta(x, shape1=true.beta.shape1[p.ind], shape2=true.beta.
              shape2[p.ind]), lty=5, col="blue", add=TRUE);
121       dev.off();
122       cat(paste("Histogram of ", p.ind, "is generated\n\n"));
123     }
124   }
125
126   # Graphical Analysis (3D, Contour Plot and Surface)
127   graphic.plot <- function(){
128     library(MASS);
129     postscript(file="beta.wl.plot.eps");
130
131     contour.level <- with(monte.space, kde2d(x1, x2));
132     with(monte.space, plot(x1, x2, pch="+", main=""));
133     contour(contour.level, add=TRUE);
134     dev.off();
135     cat("Plot is generated\n");
136
137     postscript(file="beta.wl.surface.eps");
138     persp(contour.level, phi=5, theta=40, xlab="x1", ylab="x2", zlab="
              density", main="");
139     dev.off();
140     cat("Perspective Plot is generated\n");
141   }
142
143   # Printing Numerical and Graphical Analysis
144   if(n.analysis=="TRUE"){
145
146     print(result<-numeric.analysis());
147
148     mu <- 0;
149     sd <- 0;
150     for(ind in 1:2){
151
152       mu[ind] <- (alpha[ind])/(alpha[ind]+beta[ind]);
153       sd[ind] <- sqrt( (alpha[ind]*beta[ind])/((alpha[ind]+beta[ind]+1)*
              (alpha[ind]+beta[ind])^2) );
154     }
155
156   # Calculating differences
157     for(ind in 1:2){
158       cat("Theoretical TRUE Mean and Standard Deviation\n");
159       cat(paste("MU[", ind, "] : ", mu[ind], "\n"));
160       cat(paste("SD[", ind, "] : ", sd[ind], "\n"));
161
162       cat(paste("Absolute Value of Difference between theoretical
              solution and simulation result\n"));
163       cat(paste("DIFFM", ind, " = ", abs(result$MEAN[ind]-mu[ind]), "\n"
              ));
164       cat(paste("DIFFSD", ind," = ", abs(result$SD[ind]-sd[ind]), "\n\n"
              ));
165     }
166   }
167
```

```
168   if(g.hist=="TRUE"){
169     graphic.hist();
170   }
171
172   if(g.plot == "TRUE"){
173     graphic.plot();
174   }
175
176   cat(paste("The program is terminated at ", Sys.time(),"\n\n" ));
177 }
178
179 # Simulation on July 15, 2009, Final thesis submission
180 wanglee(n.discrete.pnts=1e7, n.cnts=1e5, n=1e3, n.analysis="TRUE", g.
      hist="TRUE", g.plot="TRUE");
```

A.2 Multimodal Example

```
1 # Filename : mm.mix.wl.R [Final Thesis Submission Program]
2 # Date: 2009.07.15
3 # Programmer : Chel Hee Lee
4 # Example : Three-Components Normal Mixture Model Density
5 # Paper : Stochastic Approximation in Monte Carlo Computation, Faming
      Liang 2007
6 # Type : Wang-Lee Algorithm, Unbounded Case.
7
8 rm(list=ls(all.names=TRUE));
9
10 # Wang-Lee Algorithm Definition
11 wanglee <- function(n.discrete.pnts=1e7, n.cnts=1e5, n=1e3, last.weight
      =0.5, n.analysis="TRUE", g.hist="TRUE", g.plot="TRUE", seed
      =584479233, ... ){
12
13   cat(" *****   Program Description   ***** \n");
14   cat("Filename : mm.mix.wl.R \n");
15   cat("Description : Three-Component Mixture Model Density with Wang-Lee
        Algorithm\n\n");
16   cat("Affiliation : Statistics, University of Manitoba \n");
17   cat("Supervisor  : Dr. Liqun Wang, wangl1@cc.umanitoba.ca \n");
18   cat("Programmer  : Chel Hee Lee, umlee@cc.umanitoba.ca, gnustats@gmail
       .com \n\n");
19
20   cat(paste("Program Launching Time : ", Sys.time(),"\n\n" ));
21
22   # Random Number Specification
23   set.seed(seed);
24   cat(paste("Random Seed = ", seed, "\n\n"));
25
26   # Given parameters
27   mu <- list(c(-8,-8), c(6,6), c(0,0));
28   cov <- list(matrix(c(1,0.9,0.9,1), ncol=2), matrix(c(1,-0.9,-0.9,1),
       ncol=2), matrix(c(1, 0, 0, 1), ncol=2));
29   w <- c(1/3, 1/3, 1/3);
30
31   # Forming a discretized base
32   x <- runif(n.discrete.pnts, -12, 11);
33   y <- runif(n.discrete.pnts, -12, 11);
34
35   cat("*****   Simulation Configuration   *****\n");
36   cat(paste("Number of Discrete Base Points = ", n.discrete.pnts, "\n"))
       ;
37   cat(paste("Number of Partitions = ", n.cnts, "\n"));
38   cat(paste("Weight on Last Partition = ", last.weight, "\n"));
39   cat(paste("A Size of Sample drawn = ", n, "\n\n"));
40
41   cat("Given Values on Parameters \n");
42   for( ind in 1:3){
43     cat(paste("mu", ind, " = ", mu[[ind]], ", cov", ind, " = ", cov[[ind
         ]], ", w", ind, " = ", w[[ind]], "\n"));
```

```
44  }
45  cat("\n");
46
47  # Defining the target distribution
48  library(mnormt);
49  density <- 0;
50  for(ind in 1:3){
51    density <- density + w[ind]*dmnorm(cbind(x,y), mean=mu[[ind]],
        varcov=cov[[ind]]);
52  }
53
54  # Compact Region
55  monte.space <- data.frame(x, y, density=density);
56  rm(list=c("x", "y"));
57
58  cat(paste("Discretization procedure is completed at ", Sys.time(),"\n\
      n" ));
59
60  cat("***** Trimming Information ***** \n");
61  density <- density[ which(density > 0) ];
62  cat(paste("Number of trimmed zeros in the density = ", n.discrete.pnts
      -length(density), "\n"));
63  cat(paste("Number of remained discrete base points = ", length(density
      ), " Remaining Rate = (", 1 - (n.discrete.pnts-length(density))/
      length(density),")\n\n"));
64
65  cat("***** Quantiles of Density After Trimming ******\n");
66  print(summary(density));
67  chk.quantile <- seq(0.1, 1, 0.1);
68  print(quantile(density, chk.quantile));
69  cat("\n\n");
70
71  # Sorting by decending order
72  density.ind <- order(density, decreasing=TRUE);
73
74  # Contourization
75  lebergue.measure <- hist(density, breaks=seq(from=min(density), to=max
      (density), length.out=n.cnts+1), plot=FALSE);
76
77  # CDF
78  get.pdf <- rev(lebergue.measure$counts * lebergue.measure$mids);
79  get.pdf[n.cnts] <- last.weight*get.pdf[n.cnts];
80  nmlzd.cdf <- c(0, cumsum(get.pdf)/sum(get.pdf));
81
82  # Sampling
83  rnd.variates = runif(n);
84  p.sample.size <- hist(rnd.variates, breaks=nmlzd.cdf, plot=FALSE)$
      counts;
85  valid.samples <- which(p.sample.size >0);
86
87  cat("*****   Analysis of samples   ***** \n");
88  cat(paste("Number of Partitions used for sampling = ", length(valid.
      samples), "\n\n"));
89
```

```
90  # Searching indices from the sample space
91  sample.cnt <- mapply(sample, MoreArgs=list(replace=TRUE), rev(lebergue
        .measure$counts), p.sample.size);
92  cum.pnts.ind <- c(0, cumsum(rev(lebergue.measure$counts)));
93  cum.pnts.ind <- cum.pnts.ind[-(n.cnts+1)];
94  sample.list <- mapply("+", as.list(cum.pnts.ind), sample.cnt);
95  sample.ind <- unlist(sample.list);
96  rm(list=c("lebergue.measure", "p.sample.size", "rnd.variates", "sample
        .cnt", "cum.pnts.ind", "sample.list"));
97
98  # Saving the final samples
99  monte.space <- monte.space[density.ind[sample.ind],];
100
101 cat(paste("Sampling procedure is completed at ", Sys.time(),"\n\n" ));
102
103 # Numerical Analysis
104 numeric.analysis <- function(){
105   cat("***** Numerical Aanlysis *****\n");
106   meanMarginal <- sapply(monte.space, mean);
107   sdMarginal <- sqrt( sapply(monte.space, var) );
108   modeMarginal <- monte.space[1:5,];
109   return(list(MEAN=meanMarginal, SD=sdMarginal, MODES=modeMarginal));
110 }
111
112 # Graphical Analysis (Histogram)
113 graphic.hist <- function(){
114   parameters <- names(monte.space);
115
116   for(p.ind in parameters){
117     cat(paste("Generating histogram of parameter ", p.ind, "\n"));
118     graphic.cmd <- gsub(" ", "", paste("postscript(file='nm.mix.wl.",
          p.ind, ".eps')"));
119     eval(parse(text=graphic.cmd));
120     hist.cmd <- gsub(" ", "", paste("with(monte.space, hist(", p.ind,
          ", freq=FALSE, breaks=30, xlab=expression(", p.ind, "), main
          =''))"));
121     eval(parse(text=hist.cmd));
122     dev.off();
123     cat(paste("Histogram of ", p.ind, "is generated\n\n"));
124   }
125 }
126 # Graphical Analysis (3D, Contour Plot and Surface)
127 graphic.plot <- function(){
128   library(MASS);
129   postscript(file="nm.mix.wl.plot.eps");
130   contour.level <- with(monte.space, kde2d(x, y));
131   with(monte.space, plot(x, y, pch="+", main=""));
132   contour(contour.level, add=TRUE);
133   dev.off();
134   cat("Plot is generated\n");
135
136   postscript(file="nm.mix.wl.surface.eps");
137   persp(contour.level, phi=50, theta=25, xlab="x", ylab="y", zlab="
        density", main="");
```

```
138     dev.off();
139     cat("Perspective Plot is generated\n");
140   }
141
142   # Printing Numerical and Graphical Analysis
143   if(n.analysis=="TRUE"){
144     print(results <- numeric.analysis());
145   }
146
147   if(g.hist=="TRUE"){
148     graphic.hist();
149   }
150
151   if(g.plot == "TRUE"){
152     graphic.plot();
153   }
154
155   cat(paste("The program is terminated at ", Sys.time(),"\n\n" ));
156 }
157
158 # Simulation on July 15, 2009, Final thesis submission
159 wanglee(n.discrete.pnts=1e7, n.cnts=1e5, n=1e3, n.analysis="TRUE", g.
      hist="TRUE", g.plot="TRUE");
```

A.3 Analysis of Simulated Data

```
1 # Filename : sim.wl.R [Final Thesis Submission Program]
2 # Date: 2009.07.15
3 # Programmer : Chel Hee Lee
4 # Example : Simulated Data Analysis
5 # Paper : A Practical Sampling Approach for a Bayesian Mixture Model
      with unknown number of components, Wang & Fu 2007
6 # Type : Wang-Lee Algorithm, Unbounded Case, Weight on the last
      partition = 0.01
7
8 rm(list=ls(all.names=TRUE));
9
10 # simdata1.dat from Dr. Liqun Wang is obtained.
11 y <- c(
12 3.1915, 2.8172, 1.8725, 2.506, 2.5985, 1.7248, 5.174, 2.7769, 3.1158,
      2.9531, 3.861, 5.7123, 8.4547, 5.7908, 5.6474, 5.8743, 6.1627,
13 1.7415, 4.3249, 3.0679, 2.4846, 3.9713, 2.4585, 4.873, 3.4631, 1.7466,
      4.4115, 2.4463, 6.0181, 6.3941, 5.0767, 6.24, 5.2174, 6.0573,
14 3.7421, 1.0362, 3.2593, 1.7827, 3.8634, 1.7535, 2.8219, 3.2525, 1.6263,
      4.4871, 3.351, 6.9188, 5.2242, 6.0306, 6.5917, 6.0615, 10.2171,
15 2.4285, 2.208, 5.4172, 3.6766, 1.5599, 2.9871, 2.7607, 1.1206, 4.4464,
      3.3576, 2.2471, 5.0953, 6.9116, 5.7103, 7.8891, 6.4244, 11.1226,
16 3.9134, 1.2488, 4.9344, 2.0437, 3.9133, 2.4061, 2.0549, 2.2965, 1.831,
      2.8951, 2.2239, 5.405, 5.9909, 6.4078, 5.731, 5.265, 9.4228,
17 2.1151, 3.3802, 2.1846, 2.6896, 2.1516, 1.6863, 2.6114, 2.5205, 3.4408,
      0.9359, 7.0336, 6.5152, 5.2028, 6.5696, 6.3673, 6.9836, 8.8344,
18 3.9715, 2.8945, 3.0672, 3.8539, 1.7031, 4.8339, 3.9571, 3.8157, 3.462,
      2.8373, 6.7621, 5.1288, 7.6193, 4.939, 5.0319, 4.7349, 8.7945,
19 3.5875, 3.7998, 3.0324, 3.5942, 3.6814, 2.883, 2.8602, 2.9211, 2.9814,
      4.6747, 7.4489, 6.2629, 6.4936, 5.7563, 4.7399, 5.7748, 10.2841,
20 2.8289, 1.8612, 3.5162, 3.8315, 3.45, 2.187, 3.7803, 2.803, 3.0786,
      4.4198, 6.4474, 5.1275, 5.0068, 6.3206, 6.5225, 5.3424,
21 4.1786, 2.0207, 4.0041, 4.1895, 3.8442, 1.3753, 2.7833, 2.9003, 4.3425,
      2.0897, 6.1193, 6.7926, 5.8599, 4.6238, 4.6644, 7.7007,
22 2.593, 3.1973, 2.9382, 3.4159, 2.2244, 3.873, 3.2114, 2.7942, 1.825,
      4.6523, 7.3492, 5.9558, 6.091, 5.02, 6.9515, 7.0805,
23 2.8709, 4.3748, 3.4252, 3.6591, 2.4982, 3.7074, 3.5931, 3.3729, 3.1594,
      0.1834, 6.4623, 5.3437, 6.2259, 5.3119, 6.9811, 4.3184
24 );
25
26 wl.base <- function(data=y, n.d.pnts.cmp=2.5e6, n.cmps=4, n.cnts=1e5, n
      =1e3, prior.k=c(0.25, 0.25, 0.25, 0.25), seed=1905042700, ...){
27
28   cat(" *****   Program Description   ***** \n");
29   cat("Filename : sim.wl.R \n");
30   cat("Description : Bayesian Normal Mixture with equal variance and
       unknown number of components for Simulated Dataset with Wang-Lee
       Algorithm \n");
31   cat("Affiliation : Statistics, University of Manitoba \n");
32   cat("Supervisor  : Dr. Liqun Wang, wangl1@cc.umanitoba.ca \n");
33   cat("Programmer  : Chel Hee Lee, umlee@cc.umanitoba.ca, gnustats@gmail
       .com \n\n");
```

```
34
35  cat(paste("Program Launching Time : ", Sys.time(),"\n\n" ));
36
37  cat( "\n***** Data Analysis *****\n\n" );
38  data.length <- length(data);
39  data.min <- min(data);
40  data.max <- max(data);
41  data.range <- data.max - data.min;
42  data.sum <- sum(data);
43  data.SS <- sum(data^2);
44
45  cat( paste("Minimum = ", data.min, ", Maximum = ", data.max, " Range =
        ", data.range, "\n\n") );
46
47  cat( "Set Hyper parameters from Data Analysis.\n" );
48  h.mu <- (data.min + data.max)/2;
49  h.var <- data.range^2;
50  h.alpha <- 2;
51  h.beta <- ceiling((data.range/6)^2);
52
53  cat( paste("Hyper Mu = ", h.mu, ", Hyper Variance = ", h.var, "\n") );
54  cat( paste("Alpha = ", h.alpha, " Beta = ", h.beta, "\n\n") );
55
56  mu.min <- data.min;
57  mu.max <- data.max;
58  mu.range <- mu.max - mu.min;
59
60  var.min <- 0.1;
61  var.max <- 5;
62  var.range <- var.max - var.min;
63
64  w.min <- 0;
65  w.max <- 1;
66  w.range <- w.max - w.min;
67
68  set.seed(seed);
69  cat(paste("Random Seed = ", seed, "\n"));
70  cat(paste("Number of Mixture Components = ", n.cmps, "\n"));
71  cat(paste("Mean: Lower Limit = ", mu.min, ", Upper Limit = ", mu.max,
        "\n"));
72  cat(paste("Variance : Lower Limit = ", var.min, ",  Upper Limit = ",
        var.max, "\n"));
73  cat(paste("Weight : Lower Limit = ", w.min, ", Upper Limit = ", w.max,
        "\n\n"));
74  cat(paste("Total discrete base points(n.d.pnts) = ", n.cmps * n.d.pnts
        .cmp, "\n"));
75  cat(paste("Numbe of contours = ", n.cnts, "\n"));
76  cat(paste("\n Prior for Number of Components \n"));
77  print(prior.k);
78  cat(paste("Size of samples drawn = ", n, "\n"));
79
80  cat(paste("Data Analysis and Initial Values are set up at ", Sys.time
        (), "\n\n"));
81
```

```
82   cat("Now the program is starting a discretization\n\n");
83
84 # Defintion of Discrete Base Point Generator
85   generator <- function(add.pnts){
86
87     # Generating Random Discretized Points
88     mu.on.k <- matrix(runif(add.pnts*ind.cmp, mu.min, mu.max), ncol=ind.
         cmp);
89     if(ind.cmp != 1){          # mu1 < mu2 < mu3 < ...
90       mu.on.k <- t(apply(mu.on.k, 1, sort));
91     }
92     var.on.k <- matrix(rep(runif(add.pnts, var.min, var.max), ind.cmp),
         ncol=ind.cmp);
93     w.on.k <- log(matrix(runif(add.pnts*ind.cmp, w.min, w.max), ncol=ind
         .cmp));
94     w.on.k <- w.on.k/rowSums(w.on.k);
95
96     # Formula Setting
97     cmp.gamma <- log(ind.cmp*gamma(ind.cmp)^2) - 0.5*ind.cmp*log(2*pi*h.
         var) + log(prior.k[ind.cmp]);
98     if(ind.cmp == 1){
99       log.likelihood <- (-0.5)*data.length*log(var.on.k)-0.5/var.on.k*(
           data.SS -2*data.sum*mu.on.k + data.length*mu.on.k^2);
100      log.posterior <- log.likelihood + cmp.gamma - 0.5/h.var*(mu.on.k-h
           .mu)^2-(h.alpha+1)*log(var.on.k) - h.beta/var.on.k;
101    }
102
103    log.likelihood <- (-0.5) * data.length * log(var.on.k[,1]);
104    for( d.ind in 1:data.length){
105      norm.den <- w.on.k*exp( -0.5/var.on.k * (data[d.ind]-mu.on.k)^2 );
106      log.likelihood <- log.likelihood + log(rowSums(norm.den));
107    }
108    log.posterior <- log.likelihood + cmp.gamma - 0.5/h.var*rowSums((mu.
         on.k-h.mu)^2) - (h.alpha+1)*log(var.on.k[,1]) - h.beta/var.on.k
         [,1];
109    return(data.frame(MEAN=mu.on.k, VARIANCE=var.on.k, WEIGHT=w.on.k,
         log.likelihood=log.likelihood, log.posterior=log.posterior,
         posterior=0, I=ind.cmp));
110  }        # generator()
111
112  # Data Structure for Compact Region
113  library(R.utils);
114  monte.space <- data.frame();
115  monte.tmp <- data.frame();
116
117  for ( ind.cmp in 1:n.cmps){
118
119    s.iter <- 0;
120    add.more <- n.d.pnts.cmp;
121
122    while(add.more > 0){
123
124      s.iter <- s.iter+1;
125
```

```
126       # First generation
127       space.cmd <- gsub(" ", "", paste("s.space", ind.cmp, ".", s.iter,"
              <- generator(add.pnts=add.more);" ));
128       eval(parse(text=space.cmd));
129
130       # Inflating log.posterior to posterior
131       inflate.cmd <- gsub(" ", "", paste("s.space", ind.cmp, ".", s.iter
              ,"$posterior <- exp(s.space", ind.cmp, ".", s.iter, "$log.
              posterior);"));
132       eval(parse(text=inflate.cmd));
133
134       # Filtering in the significant region
135       filter.cmd <- gsub(" ", "", paste("s.space", ind.cmp, ".", s.iter,
              " <- s.space", ind.cmp, ".", s.iter, "[which(s.space", ind.cmp
              , ".", s.iter,"$posterior > 0), ];"));
136       eval(parse(text=filter.cmd));
137
138       more.cmd <- gsub(" ", "", paste("add.more <- add.more - nrow(s.
              space", ind.cmp, ".", s.iter ,");"));
139       eval(parse(text=more.cmd));
140     }
141
142     pnt.cmd <- gsub(" ", "", paste("space", ind.cmp, "<- data.frame()"))
          ;
143     eval(parse(text=pnt.cmd));
144
145     # Creating the sample space in the significant region
146     for( i in 1:s.iter){
147       space.cmd <- gsub(" ", "", paste("space", ind.cmp, "<- rbind(space
              ", ind.cmp, ", s.space", ind.cmp, ".", i,")"));
148       eval(parse(text=space.cmd));
149
150       rm.cmd <- gsub(" ", "", paste("rm(s.space", ind.cmp, ".", i,");"))
              ;
151       eval(parse(text=rm.cmd));
152     }
153
154     c.cmd <- gsub(" ", "", paste("n.rows <- nrow(space", ind.cmp,")"));
155     eval(parse(text=c.cmd));
156
157     cat("\n");
158     cat(paste(n.rows, " base points in ", ind.cmp,"th component are
              cumulated in the significant region\n"));
159
160     # MLE and AMLE
161     cat(paste("\n***** The MLE and AMLE in ", ind.cmp, " th component **
              *** \n"));
162     search.cmd <- gsub(" ", "", paste("max.ind <- which.max(space", ind.
              cmp,"$log.likelihood);"));
163     eval(parse(text=search.cmd));
164     mle.cmd <- gsub(" ", "", paste("space",ind.cmp,"[max.ind,];"));
165     print(eval(parse(text=mle.cmd)));
166     cat("\n ");
167
```

```
168     # Mode of Log-Posterior Estimator and its Joint mode
169     cat(paste("\n***** The Mode of log-posterior in ", ind.cmp, " th
            component ***** \n"));
170     search.cmd <- gsub(" ", "", paste("max.ind <- which.max(space", ind.
            cmp,"$posterior);"));
171     eval(parse(text=search.cmd));
172     mle.cmd <- gsub(" ", "", paste("space",ind.cmp,"[max.ind,];"));
173     print(eval(parse(text=mle.cmd)));
174     cat("\n ");
175
176     # Adding KEY index for searching and querying
177     key.cmd <- gsub(" ", "", paste("space", ind.cmp," <- data.frame(
            space", ind.cmp, ", KEY=seq(nrow(space", ind.cmp,")));" ));
178     eval(parse(text=key.cmd));
179
180     # Saving Objects
181     size.cmd <- gsub(" ", "", paste("object.size(space",ind.cmp,");"));
182     cat(paste("Object size of space ", ind.cmp, " is ", eval(parse(text=
            size.cmd)), "\n" ));
183     save.cmd <- gsub(" ", "", paste("saveObject(space", ind.cmp, ", file
            ='space", ind.cmp, ".RData');") );
184     eval(parse(text=save.cmd));
185
186     # Creating Monte Compact Space
187     monte.cmd <- gsub(" ", "", paste("monte.tmp", ind.cmp," <- data.
            frame(evaluates=space", ind.cmp,"$log.posterior, cmp.id=space",
            ind.cmp, "$I, KEY=space", ind.cmp, "$KEY);"));
188     eval(parse(text=monte.cmd));
189
190     save.monte.cmd <- gsub(" ", "", paste("saveObject(monte.tmp",ind.cmp
            ,", file='monte.tmp",ind.cmp,".RData')"));
191     eval(parse(text=save.monte.cmd));
192
193     cat(paste("Object Size of Monte Space", ind.cmp," is ", object.size(
            monte.space),"\n" ));
194
195     # Removing Objects for saving memory
196     rm.cmd <- gsub(" ", "", paste("rm(space", ind.cmp,")"));
197     eval(parse(tex=rm.cmd));
198   }
199
200   cat(paste("Discretization is completed at ", Sys.time(), "\n MLE and
          AMLE are found. \n\n"));
201   rm(list=ls(all.names=TRUE));
202 }
203
204 wl.sampling.control <- function(n.cmps=4, n.cnts=1e5, n=1e3, control=1,
        p.last=0.01, ...){
205
206   library(R.utils);
207   monte.space <- data.frame();
208
209   cat(paste("\n Proportion used in last contour = ", p.last," \n\n" ));
210
```

```
211  for(ind.cmp in 1:n.cmps){
212    load.cmd <- gsub(" ", "", paste("monte.tmp",ind.cmp," <- loadObject
         ('monte.tmp", ind.cmp,".RData');" ));
213    eval(parse(text=load.cmd));
214    monte.space.cmd <- gsub(" ", "", paste("monte.space <- rbind(monte.
         space, monte.tmp",ind.cmp,");"));
215    eval(parse(text=monte.space.cmd));
216  }
217
218  n.d.pnts <- nrow(monte.space);
219  n.pnts.cnt <- n.d.pnts/n.cnts;
220
221  chk.quantile <- seq(0.1, 1, 0.1);
222
223  # Posterior Information
224  cat("***** Quantiles of Log Posterior ******\n");
225  print(summary(monte.space$evaluates));
226  print(quantile(monte.space$evaluates, chk.quantile));
227  cat("\n");
228
229  # Inflate log value to the original values (taking exponent)
230  monte.space$evaluates <- with(monte.space, exp(evaluates));
231
232  # Posterior Information
233  cat("***** Quantiles of Posterior ******\n");
234  print(summary(monte.space$evaluates));
235  print(quantile(monte.space$evaluates, chk.quantile));
236  cat("\n");
237
238  # Sorting all log-posterior points by descending order
239  monte.space <- with(monte.space, monte.space[order(evaluates,
         decreasing=TRUE),]);
240
241  # Saving mode information
242  mode.info <- monte.space[1,];
243
244  # Contourization
245  lebergue.measure <- with(monte.space, hist(evaluates, breaks=seq(from=
         min(evaluates), to=max(evaluates), length.out=n.cnts+1), plot=
         FALSE));
246
247  cat("\n***** Analysis of Contours *****\n");
248  cat(paste("Height in a contour in LOG = ", log(lebergue.measure$mids
         [2]-lebergue.measure$mids[1]),"\n"));
249  cat(paste("Height(Midpoint) in a contour = ", lebergue.measure$mids
         [2]-lebergue.measure$mids[1],"\n"));
250  cat(paste("Number of points in last contour = ", lebergue.measure$
         counts[control] ," \n"));
251  cat(paste("The proportion to be eliminated on points if neccessary = "
         , lebergue.measure$counts[control]/nrow(monte.space) ," \n"));
252
253  # Normalized CDF
254  get.pdf <- rev(lebergue.measure$counts * lebergue.measure$mids);
255
```

```
256  # Consider low probability in the tails
257  get.pdf[n.cnts] <- get.pdf[n.cnts]*p.last;
258  nmlzd.cdf <- c(0, cumsum(get.pdf)/sum(get.pdf));
259
260  # Sampling from the range covering all components
261  rnd.variates = runif(n);
262  p.sample.size <- hist(rnd.variates, breaks=nmlzd.cdf, plot=FALSE)$
         counts;
263  valid.samples <- which(p.sample.size>0);
264
265  cat("*****Analysis of samples*****\n");
266  cat(paste("The number of contours containing sample points = ", length
         (valid.samples), "\n"));
267
268  # Searching index from original sample space
269  sample.cnt <- mapply(sample, MoreArgs=list(replace=TRUE), rev(lebergue
         .measure$counts), p.sample.size);
270  cum.pnts.ind <- c(0, cumsum(rev(lebergue.measure$counts)));
271  cum.pnts.ind <- cum.pnts.ind[-(n.cnts+1)];
272  sample.list <- mapply("+", as.list(cum.pnts.ind), sample.cnt);
273  sample.ind <- unlist(sample.list);
274  monte.space <- monte.space[sample.ind, ];
275
276  cat(paste("*****Marginal Posterior Distribution of Component K*****"))
         ;
277  print(table(monte.space$cmp.id)/length(monte.space$cmp.id));
278
279  # Matching and Querying
280  query.space <- split(monte.space, monte.space$cmp.id);
281
282  for ( ind.cmp in 1:n.cmps){
283    load.cmd <- gsub(" ", "", paste("space", ind.cmp," <- loadObject('
           space",ind.cmp,".RData');"));
284    eval(parse(text=load.cmd));
285
286    if(mode.info$cmp.id == ind.cmp){
287      p.mode.cmd <- gsub(" ", "", paste("p.mode <- space",mode.info$cmp.
             id,"[mode.info$KEY, ]"));
288      eval(parse(text=p.mode.cmd));
289
290      # Mode of log-posterior and its joint mode
291      cat(paste("\n ***** Joint Mode for Log-Posterior is found at ",
             mode.info$cmp.id , "th component. ***** \n"));
292      print(p.mode);
293      cat("\n");
294    }
295
296    query.cmd <- gsub(" ", "", paste("space", ind.cmp,"<- space", ind.
           cmp, "[query.space$'", ind.cmp, "'$KEY,]"));
297    eval(parse(text=query.cmd));
298
299    # Saving samples from spaces
300    save.samples <- gsub(" ", "", paste("sample",ind.cmp," <- saveObject
           (space",ind.cmp,", file='sample",ind.cmp,".RData');"));
```

```
301     eval(parse(text=save.samples));
302   }
303   cat(paste("\nQuerying procedure for all spaces is completed.\n" ));
304   cat(paste("Sampling procedure is completed at ", Sys.time(),"\n\n" ));
305   rm(list=ls(all.names=TRUE));
306 }
307
308 wl.analysis<- function(n.cmps=4){
309   library(R.utils);
310
311   # Numerical Analysis
312   for(ind.cmp in 1:n.cmps){
313     sample.cmd <- gsub(" ", "", paste("sample.ind <- loadObject('sample"
          , ind.cmp,".RData')" ));
314     eval(parse(text=sample.cmd));
315
316     if(nrow(sample.ind)==0){
317       cat(paste("There is no drawed samples in this ", ind.cmp, " th
            component in a mixture\n"));
318     }
319     else{
320       meanMarginal <- sapply(sample.ind, mean);
321       sdMarginal <- sqrt( sapply(sample.ind, var) );
322       cat(paste("\n***** Numerical Information on ", ind.cmp, "th
            component *****\n"));
323       print(list(MEAN=meanMarginal, SD=sdMarginal));
324
325       vars <- names(sample.ind);
326       vars <- vars[-(length(vars)-3):-length(vars)]
327       print(vars);
328
329       for ( v.ind in vars) {
330  file.cmd<- gsub(" ", "", paste("postscript(file='sim.wl.K.", ind.cmp, "
       .", v.ind, ".eps');"));
331  eval(parse(text=file.cmd));
332  hist.cmd<- gsub(" ", "", paste("hist(sample.ind$", v.ind, ", freq=FALSE
       , breaks=40, main=' ', xlab=' Component.K=", ind.cmp, ".", v.ind,"
       ');"));
333  eval(parse(text=hist.cmd));
334  dev.off();
335       }
336       cat(paste("All Histograms are created in ", ind.cmp, " th
            component \n\n"));
337       rm(sample.ind);
338     }
339   }
340   cat(paste("\nNumerical Analysis for all spaces is completed at ", Sys.
        time(),"\n" ));
341   cat(paste("The program is completed at ", Sys.time(),"\n\n" ));
342   rm(list=ls(all.names=TRUE));
343 }
344
345 # Simulation on July 15, 2009, Final thesis submission
```

```
346 wl.base(data=y, n.d.pnts.cmp=2.5e6, n.cmps=4, n.cnts=1e5, n=1e3, prior.k
      =c(0.25, 0.25, 0.25, 0.25));
347 wl.sampling.control(n.cmps=4, n.cnts=1e5, n=1e3, p.last=0.01);
348 wl.analysis(n.cmps=4);
349
350 rm(list=ls(all.names=TRUE));
```