New Approaches to Importance Sampling for Portfolio Credit Risk Valuation

by

Zhe Wang

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

New Approaches to Importance Sampling for Portfolio Credit Risk Valuation

Zhe Wang
Master of Science
Graduate Department of Computer Science
University of Toronto
2015

Portfolio credit risk based on the Gaussian Copula model has been widely studied and generally evaluated through Monte Carlo simulations. The two-level structure, namely systematic factors and individual factors, complicates the problem in a way that traditional variance reduction techniques become very hard to apply. Glasserman and Li proposed a two-level importance sampling approach to tackle a simplified binary credit states problem. The inner level was approximated through a conditional importance sampling approach using an exponential twisting technique. In this research project, we propose an alternative importance sampling approach which uses the Central Limit Theorem for the inner level. Our approach can be easily generalized to multi-credit states. Based on this approximation, we then propose two novel approaches motivated from research in machine learning. Instead of finding the importance sampler through an optimization problem, the first approach approximates the zero variance function by learning from the samples which are generated from Markov Chain Monte Carlo. The second approach treats the problem as a Bayesian inference problem and evaluates the tail probability through Bayesian Monte Carlo. Compared to Glasserman and Li's method, numerical results show that these two new approaches have advantages in both accuracy and speed and are also more easily adapted to other applications.

Acknowledgements

I would not have been able to complete this project without the support from many people. Firstly, I am deeply thankful to my supervisor, Professor Ken Jackson, whose endless patience and enlightening suggestions, from the beginning to the end of this project, guided me to develop a thorough understanding of this topic. I would like to thank Dr. Alexander Kreinin, Head of the Quantitative Research, Risk Analytics, IBM, who generously contributed his valuable time to be the second reader of this research paper. I also wish to express my sincere gratitude to all my friends, especially Yanshuai Cao, for stimulating discussions and incisive advice and to Dr. Mohammad Shakourifar and Meng Han for their astute guidance. I'm also grateful to my family for their endless love. A very special thanks to my life partner, Jennifer. She always stands by me and offers unwavering support.

Contents

1 Introduction **1**

1.1 Gaussian Copula Factor Model 1

1.2 A Review of Related Work 3

1.2.1 Central Limit Theorem 3

1.2.2 Binary Credit States CLT 5

1.3 An Outline of the Thesis 6

2 Zero Variance Function Estimation **7**

2.1 Zero Variance Function 7

2.2 Cross Entropy 8

2.3 Markov Chain Monte Carlo 12

2.3.1 Metropolis Sampler 12

2.3.2 Variable-At-A-Time and Parallel Tempering MCMC 13

2.3.3 Slice Sampling 15

2.4 Gaussian Mixture Model 16

2.4.1 Expectation-Maximization 16

2.4.2 Drawbacks of EM 18

3 Bayesian Monte Carlo **19**

3.1 Bayesian Linear Model 19

3.2 Gaussian Process Model 21

3.3 Bayesian Monte Carlo 23

3.4 Drawback of BMC 25

3.5 Hybrid BMC 26

4 Two Step Importance Sampling **29**

4.1 Conditional Importance Sampling 30

4.2 Outer level Importance Sampling 31

5 Numerical Experiments **33**

5.1 Configurations 33

5.2 99.9% VaR Computation 34

5.3 Tail probability calculation 39

6 Conclusion **48**

Bibliography **48**

List of Tables

1.1 Credit Migration Matrix with 4 states. 2
1.2 Equivalent Credit Migration Matrix 5

5.1 99.9% VaR comparison for four methods given precomputed overhead tested for $S = 20$ and $N = 2500$ 36
5.2 99.9% VaR comparison for four methods given precomputed overhead tested for $S = 30$ and $N = 2500$ 37
5.3 Comparison of the numerical results for $S = 5$ 44
5.4 Running time of the methods tested for $S = 5$ 44
5.5 Comparison of the numerical results for $S = 10$ 45
5.6 Running time of the methods tested for $S = 10$ 45
5.7 Comparison of the numerical results for $S = 20$ 46
5.8 Running time of the methods tested for $S = 20$ 46
5.9 Comparison of the numerical results for $S = 30$ 47
5.10 Running time of the methods tested for $S = 30$ 47

List of Figures

2.1 1D zero variance function for various loss levels 9
2.2 2D zero variance function for various loss levels 10

3.1 A toy example for BMC in 1D 25
3.2 A toy example for BMC in 1D with better estimation 26
3.3 A toy example for a high loss level in 1D 27
3.4 A toy example for the hybrid BMC in 1D 27
3.5 A toy example showing the effects of a poor importance sampler 28
3.6 A toy example illustrating a 2 mode importance sampler 28

5.1 99.9% VaR comparison under $S = 20$ for $N = 2500$ 38
5.2 99.9% VaR comparison under $S = 30$ for $N = 2500$ 40
5.3 tail probability given the loss level l_k for dimension $S = 5$ 42
5.4 tail probability given the loss level l_k for dimension $S = 10$ 42
5.5 tail probability given the loss level l_k for dimension $S = 20$ 43
5.6 tail probability given the loss level l_k for dimension $S = 30$ 43

Chapter 1

Introduction

1.1 Gaussian Copula Factor Model

Credit risk is one of the crucial risks financial institutes need to manage. This was dramatically underscored during the sub-prime mortgage crisis. Credit risk refers to the possible loss due to default or credit rating downgrades of debtors associated with a portfolio (sometimes the debtors are also called obligors). One characteristic of credit risk is the so called "black swan event", which rarely occurs but causes significant loss to investors if it happens.

A few models have been developed to measure credit risk; the most widely used one is the Gaussian Copula Factor Model. In this framework, obligors' risks are modeled by a set of underlying risk factors known as systemic and idiosyncratic risk factors. In this section, we review this approach, following the notation in Han's research proposal [5]. The notation below is used throughout this paper:

$$
\begin{aligned}
N &= \text{number of obligors} \\
C &= \text{number of credit states, numbered } \{0, \ldots, C-1\} \\
S &= \text{dimension of system risk factor} \\
\mathcal{L} &= \text{the percentage loss of the portfolio} \\
\mathbb{1}_n^c &= \text{indicator function: 1 if obligor } n \text{ is in credit state } c \text{ and 0 otherwise} \\
EAD_n &= \text{exposure-at-default, i.e., the value that is lost if obligor } n \text{ defaults} \\
LGC_n^c &= \text{percentage loss/gain if obligor } n \text{ moves to credit state } c \\
\mathcal{Y}_n &= \text{creditworthiness index of obligor } n \\
P_\eta^\gamma &= \text{the probability of moving from credit state } \eta \text{ to credit state } \gamma \\
c(n) &= \text{the initial credit state for obligor } n
\end{aligned}
$$

The percentage loss of the total portfolio can be expressed as

$$
\mathcal{L} = \frac{\sum_{n=1}^{N} EAD_n (\sum_{c=0}^{C-1} LGC_n^c \mathbb{1}_n^c)}{\sum_{n=1}^{N} EAD_n} = \sum_{n=1}^{N} \omega_n \left(\sum_{c=0}^{C-1} LGC_n^c \mathbb{1}_n^c \right) = \sum_{n=1}^{N} \omega_n \mathcal{L}_n \tag{1.1}
$$

where $\omega_n = \frac{EAD_n}{\sum_{n=1}^{N} EAD_n}$ is the exposure weight of obligor n and $\mathcal{L}_n = \sum_{c=0}^{C-1} LGC_n^c \mathbb{1}_n^c$ is the loss rate of

obligor n. Therefore $\mathcal{L}$ can be expressed as an affine combination of $\mathcal{L}_n$. Note that $\mathcal{L}$ can be positive, negative or zero, where a negative loss is a gain.

c1/c2	A(3)	B(2)	C(1)	D(0)
A(3)	0.9796	0.0202	0.0000	0.0002
B(2)	0.0208	0.9398	0.0125	0.0270
C(1)	0.0000	0.0649	0.6801	0.2550
D(0)	0.0000	0.0000	0.0000	1.0000

Table 1.1: Credit Migration Matrix with 4 states. State A is numbered 3 and the default state, D, is numbered 0. Each cell gives the probability of moving from the credit state associated with its row to the credit state associated with its column. This table is from Han's Research Proposal [5].

In Merton's default model[7], a credit event is modeled as

$$\mathbb{1}_n^c = \mathbb{1}_{\{H_{c(n)}^{c-1} \le \mathcal{Y}_n \le H_{c(n)}^c\}} \quad \text{and} \quad H_{c(n)}^c = \Phi^{-1}\left(\sum_{\gamma \le c} P_{c(n)}^{\gamma}\right) \tag{1.2}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function and P_η^γ is the probability that the credit state migrates from state η to state γ. As an illustration, values for P_η^γ are given in Table 1.1.

In the above equation, $H_{c(n)}^c$ can be regarded as the threshold for the creditworthiness index if the credit rating of obligor n migrates from the initial state $c(n)$ to state c. Typically this creditworthiness index $\mathcal{Y}_n$ is further decomposed into two parts, systemic and non-systemic (also called idiosyncratic) risk, such that

$$\mathcal{Y}_n = \beta_n^T \mathcal{Z} + \sqrt{1 - \beta_n^T \beta_n}\varepsilon_n \tag{1.3}$$

where $\mathcal{Z} = (Z_1, \ldots, Z_S)^T$ is the independent systemic risk factor with dimension S, $\beta_n = (\beta_{1n}, \ldots, \beta_{Sn})^T$ for which each component β_{in} represents obligor n's sensitivity to the i^{th} component of the systemic risk factor, and $\mathcal{E} = (\varepsilon_1, \ldots, \varepsilon_N)$ is the independent individual idiosyncratic risk. In this paper, we assume that $\mathcal{Z}$ follows the normal distribution $N(0, I_S)$ and $\mathcal{E}$ follows the normal distribution $N(0, I_N)$ and $\mathcal{Z}$ and $\mathcal{E}$ are independent. With creditworthiness defined in equation (1.3), one can rewrite the credit event indicator (1.2) as

$$\mathbb{1}_n^c = \mathbb{1}_{\left\{\frac{H_{c(n)}^{c-1} - \beta_n^T \mathcal{Z}}{\sqrt{1-\beta_n^T\beta_n}} \le \varepsilon_n < \frac{H_{c(n)}^{c} - \beta_n^T \mathcal{Z}}{\sqrt{1-\beta_n^T\beta_n}}\right\}} \tag{1.4}$$

Therefore with a specific systemic risk factor $\mathcal{Z}$ and individual risks $\mathcal{E}$ given, the total percentage loss can be calculated under this scenario:

$$\begin{aligned}\mathcal{L} = L_N(\mathcal{Z}, \mathcal{E}) &= \sum_{n=1}^N \omega_n \mathcal{L}_n = \sum_{n=1}^N \omega_n \left(\sum_{c=0}^{C-1} LGC_n^c \mathbb{1}_n^c\right) \\ &= \sum_{n=1}^N \omega_n \left(\sum_{c=0}^{C-1} LGC_n^c \mathbb{1}_{\left\{\frac{H_{c(n)}^{c-1} - \beta_n^T \mathcal{Z}}{\sqrt{1-\beta_n^T\beta_n}} \le \varepsilon_n < \frac{H_{c(n)}^{c} - \beta_n^T \mathcal{Z}}{\sqrt{1-\beta_n^T\beta_n}}\right\}}\right)\end{aligned} \tag{1.5}$$

The loss probability of a portfolio given a quantile l is given by

$$\mathbb{P}\{L_N(\mathcal{Z}, \mathcal{E}) > l\} = \mathbb{E}[\mathbb{P}\{L_N(\mathcal{Z}, \mathcal{E}) > l | \mathcal{Z} = \mathbf{z}\}] = \int_{\mathbb{R}^S} \mathbb{P}\{L_N(\mathbf{z}, \mathcal{E}) > l\} d\Phi_S(\mathbf{z}) \tag{1.6}$$

1.2 A Review of Related Work

The Gaussian Copula risk factor model gives a way to measure the loss probability stated in equation (1.6). There are several ways to evaluate the integral. The naive approach is to use a 2-level crude Monte Carlo (MC): in an outer loop, one samples the systemic risk factor $\mathcal{Z}$ first; then conditional on this particular sample $\mathbf{z}$, in an inner loop, one samples $\mathcal{E}$ and evaluates equation (1.5) repeatedly. The accuracy of this two-level MC simulation heavily depends on the number of samples. That is to say, this approach becomes extremely time consuming if we want to achieve an accurate result. We review two related pieces of work. The first one, proposed by Han [5], replaces the inner level of the 2-level MC simulation by an analytic approximation thereby reducing the 2-level MC simulation to a 1-level MC simulation, thus achieving a significant speed-up. The second approach, proposed by Glasserman and Li [3], uses importance sampling to speed up the 2-level MC simulation. We postpone reviewing the Glasserman and Li approach to Chapter 4 and use it as a benchmark for our new algorithms described in Chapters 2 and 3.

1.2.1 Central Limit Theorem

Gordy suggested an asymptotic approximation of $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\}$ based on the law of large numbers (LLN) [4]. Conditioning on $\mathcal{Z} = \mathbf{z}$, the loss converges to its expectation as N increases:

$$L_N(\mathbf{z},\mathcal{E}) \xrightarrow{a.s.} \mathbb{E}[L_N(\mathbf{z},\mathcal{E})], \quad \text{as} \quad N \to \infty \tag{1.7}$$

Hence denoting $\mathbb{E}[L_N(\mathbf{z},\mathcal{E})]$ by $\mu_N(\mathbf{z})$, the conditional loss probability is

$$\mathbb{P}\{L_N(\mathbf{z},\mathcal{E}) > l\} \approx \mathbb{1}_{\{\mu_N(\mathbf{z})>l\}} \tag{1.8}$$

This approximation saves a significant amount of computational work by replacing the inner MC simulation by an asymptotic approximation. Furthermore, the assumption that N is large is reasonable in practice for financial institutions, as most of the portfolios they hold contain enough obligors to achieve risk diversification. However, for coarse-grained or heterogeneous portfolios, this approach may be inaccurate [6].

Based on Gordy's work, Han [5] proposed a better approximation based on the central limit theorem (CLT). The following theorem is a key result in his paper.

Theorem 1.1. *Conditional on $\mathcal{Z} = \mathbf{z}$, if the following conditions hold*

1. *$\exists \delta > 0$ such that $\sup_n\{|\omega_n|\} = \mathcal{O}(N^{-(1/2+\delta)})$,*
2. *$\exists M \in [0, +\infty)$ such that $\mathbf{z} \in \mathcal{D}^S = [-M, M] \times \cdots \times [-M, M]$,*

then the normalized conditional portfolio loss converges in distribution to a standard normal random variable:

$$\frac{L_N(\mathbf{z},\mathcal{E}) - \mu_N(\mathbf{z})}{\sigma_N(\mathbf{z})} \xrightarrow{d} \mathcal{N}(0,1)$$

as $N \to \infty$, where $\mu_N(\mathbf{z}) = \mathbb{E}[L_N(\mathbf{z},\mathcal{E})]$ and $\sigma_N^2(\mathbf{z}) = \mathbb{V}[L_N(\mathbf{z},\mathcal{E})]$.

Condition 1 in the above theorem guarantees that as $N \to \infty$, the largest weight in the portfolio decays at a rate of $\mathcal{O}(N^{-(1/2+\delta)})$. Condition 2 ensures that the systemic risk factor region is compact.

By taking expectation on both sides of equation (1.5), one obtains

$$\begin{aligned}\mu_N(\mathbf{z}) &= \mathbb{E}[L_N(\mathcal{Z},\mathcal{E})|\mathcal{Z}=\mathbf{z}] \\ &= \sum_{n=1}^{N}\omega_n\left(\sum_{c=0}^{C-1} LGC_n^c \mathbb{E}\left[\mathbb{1}_{\left\{\frac{H_{c(n)}^{c-1}-\beta_n^T\mathcal{Z}}{\sqrt{1-\beta_n^T\beta_n}}\le\varepsilon_n<\frac{H_{c(n)}^{c}-\beta_n^T\mathcal{Z}}{\sqrt{1-\beta_n^T\beta_n}}\right\}}\right]\right) \\ &= \sum_{n=1}^{N}\omega_n\left(\sum_{c=0}^{C-1} LGC_n^c\left(\Phi\left(\frac{H_{c(n)}^{c}-\beta_n^T\mathbf{z}}{\sqrt{1-\beta_n^T\beta_n}}\right)-\Phi\left(\frac{H_{c(n)}^{c-1}-\beta_n^T\mathbf{z}}{\sqrt{1-\beta_n^T\beta_n}}\right)\right)\right) \\ &= \sum_{n=1}^{N}\omega_n\left(\sum_{c=0}^{C-1} LGC_n^c p_n^c(\mathbf{z})\right) \end{aligned} \tag{1.9}$$

where

$$p_n^c(\mathbf{z}) = \Phi\left(\frac{H_{c(n)}^{c}-\beta_n^T\mathbf{z}}{\sqrt{1-\beta_n^T\beta_n}}\right)-\Phi\left(\frac{H_{c(n)}^{c-1}-\beta_n^T\mathbf{z}}{\sqrt{1-\beta_n^T\beta_n}}\right)\in[0,1] \tag{1.10}$$

and

$$\begin{aligned}\sigma_N^2(\mathbf{z}) &= \mathbb{V}[L_N(\mathbf{z},\mathcal{E})] \\ &= \sum_{n=1}^{N}\omega_n^2\left(\frac{1}{2}\sum_{a=0}^{C-1}\sum_{b=0}^{C-1}(LGC_n^a-LGC_n^b)^2 p_n^a(\mathbf{z})p_n^b(\mathbf{z})\right) \\ &= \sum_{n=1}^{N}\omega_n^2\left(\sum_{a>b}(LGC_n^a-LGC_n^b)^2 p_n^a(\mathbf{z})p_n^b(\mathbf{z})\right) \end{aligned} \tag{1.11}$$

Note that $p_n^c(\mathbf{z})$ is the conditional probability that obligor n moves from its current credit state to credit state c. In particular, when $c=0$, $\mathbb{1}_n^0(\mathbf{z})$ is the conditional default indicator for obligor n and $p_n^0(\mathbf{z}) = \mathbb{E}[\mathbb{1}_n^0(\mathbf{z})]$ is therefore the conditional default probability. We introduce the notation $\mathbb{1}_n^D(\mathbf{z}) \equiv \mathbb{1}_n^0(\mathbf{z})$ and $p_n^D(\mathbf{z}) \equiv p_n^0(\mathbf{z})$ that these are the default indicators and default probabilities, respectively. Interested readers are referred to Han [5] for a detailed proof of Theorem 1.1. With the mean and variance defined above, the conditional loss probability is given by

$$\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E})>l|\mathcal{Z}=\mathbf{z}\}\approx 1-\Phi\left(\frac{l-\mu_N(\mathbf{z})}{\sigma_N(\mathbf{z})}\right) \tag{1.12}$$

and the total portfolio loss is

$$\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E})>l\}\approx\int_{\mathbb{R}^S}\left(1-\Phi\left(\frac{l-\mu_N(\mathbf{z})}{\sigma_N(\mathbf{z})}\right)\right)d\Phi_S(\mathbf{z}) \tag{1.13}$$

The goal of this paper is to approximate the total portfolio loss by developing an effective scheme to approximate the integral on the right side of equation (1.13). If S is small, we can approximate this integral by quadrature. However, in many applications, S is in the range 10 to 30. For such S, quadrature is not a viable option. In this paper, we explore variance reduction methods for Monte Carlo approximation of this integration.

1.2.2 Binary Credit States CLT

If we look at just one industry, it is reasonable to assume that credit ratings of obligors in that industry transit following a credit migration matrix such as the one in Table 1.1. If obligors come from different industries, it may be more realistic to augment the credit migration matrix so that different transition probabilities are allowed for different industries.

We first look at a simplified case - binary credit states. That is, $C = 1$ and there are only two credit states 0 and 1 where 0 is the default state D. Suppose obligors in the portfolio come from different industries. Let p_n^D be the probability that obligor n migrates to the default state D. The credit migration matrix can be altered by assuming that obligor n is in the non default credit state $c(n) = 1$. The obligor's credit state either remains unchanged, wth probability $1 - p_n^D$, or jumps to the default state, with probability p_n^D. Table 1.2 provides an example of a credit migration matrix for this model. In this case, equation (1.2) simplifies to

$$H_{c(n)}^D = \Phi^{-1}(1 - p_n^D).$$

The conditional default probability in equation (1.10) is thus

$$p_n^D(\mathbf{z}) = 1 - \Phi\left(\frac{\Phi^{-1}(1 - p_n^D) - \beta_n^T \mathbf{z}}{\sqrt{1 - \beta_n^T \beta_n}}\right) = \Phi\left(\frac{\Phi^{-1}(p_n^D) + \beta_n^T \mathbf{z}}{\sqrt{1 - \beta_n^T \beta_n}}\right)$$

As a result, the conditional mean and variance of the loss are

$$\mu_N(\mathbf{z}) = \sum_{n=1}^{N} h_n p_n^D(\mathbf{z}), \qquad \sigma_N^2(\mathbf{z}) = \sum_{n=1}^{N} h_n^2 p_n^D(\mathbf{z})(1 - p_n^D(\mathbf{z}))$$

where $h_n = \omega_n LGC_n^D$, the weighted loss percentage at default for obligor n.

c1/c2	c(1)	c(2)	$\cdots$	c(N)	D
c(1)	$1 - p_1^D$	0	$\cdots$	0	p_1^D
c(2)	0	$1 - p_2^D$	$\cdots$	0	p_2^D
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
c(N)	0	0	$\cdots$	$1 - p_N^D$	p_N^D
D	0	0	0	0	1

Table 1.2: Credit Migration Matrix for which each credit state is associated with a particular obligor. Obligor n's credit rating either remains unchanged, with probability $1 - p_n^D$, or jumps to the default state, with probability p_n^D.

The binary-credit-state simplification removes many intermediate credit states that are in the multi-credit-state model. Under a multi-credit-state framework, a credit event may trigger a minor credit state migration in Table 1.1 and a small portfolio loss. However, in the simplified binary-credit-state framework, the same credit event may either cause a significant loss, or not change the portfolio loss level at all. Thus, the risk factor surface is less smooth for the binary-credit-state model than for the multi-credit-states model.

To incorporate multi-credit-states into a cross industry framework, one can replace each nonzero cell in Table 1.2 with a sub-matrix similar to Table 1.1 and merge default states from all the industries into

the last state.

1.3 An Outline of the Thesis

We proposed two approaches to approximate the portfolio loss function $\mathbb{P}\{L_N(\mathcal{Z}, \mathcal{E}) > l\}$. The first approach is based on importance sampling and approximates the zero-variance function (explained in Chapter 2) by a Gaussian Mixture Model. The other approach treats the integration (1.13) as a Bayesian inference problem, where a Gaussian Process and a Bayesian quadrature would be adopted.

The first approach is investigated in Chapter 2. This chapter is further subdivided into two parts. Firstly, we focus on the zero variance function sampling. Such an idea comes from an improved cross entropy method. The Markov Chain Monte Carlo (MCMC) sampling techniques are used and two common modified versions (Variable-At-A-Time MCMC and Parallel Tempered MCMC) in high dimension sampling are investigated. We conclude MCMC sampling by looking at another technique, called slice sampling, which overcomes the flaws from both Variable-At-A-Time and Parallel Tempered MCMC. Samples from MCMC are assumed to be from Gaussian Mixture Models (GMM). In the second part of Chapter 2, we explore GMM and estimate the zero-variance function through the Expectation-Maximization (EM) algorithm. However, a disadvantage of GMM is that it uses a predetermined number of Gaussian components, which must be determined a priori. In Chapter 3, we look at an entirely different approach to evaluate the integral in equation (1.13), called Bayesian Monte Carlo (BMC). This sampling method requires the knowledge of Gaussian process modeling, which we review in Chapter 3.

In Chapter 4, we give a brief review of Glasserman and Li's [3] two-step important sampling approach. In Chapter 5, we compare numerical results for three methods: Glasserman and Li's two level importance sampling, MCMC with the Gaussian Mixture Model, and the Bayesian Monte Carlo method. In Chapter 6, we summarize the results and discuss future work.

Chapter 2

Zero Variance Function Estimation

2.1 Zero Variance Function

As explained in section 1.2.1, using the Central Limit Theorem, we are able to remove one level of sampling and look for some unconditional sampling techniques to evaluate

$$\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\} \approx \int_{\mathbb{R}^S} \left(1 - \Phi\left(\frac{l - \mu_n(\mathbf{z})}{\sigma_N(\mathbf{z})}\right)\right) d\Phi_S(\mathbf{z}) \tag{2.1}$$

where Φ is the cdf of a standard normal random variable and $\mu_n(\mathbf{z})$ and $\sigma_N(\mathbf{z})$ are given by (1.9) and (1.11), respectively. Consider importance sampling

$$I \equiv \mathbb{E}_f[h(Y)] = \int h(y)f(y)dy = \int h(x)\frac{f(x)}{\pi(x)}\pi(x)dx = \mathbb{E}_\pi\left[h(X)\frac{f(X)}{\pi(X)}\right] \tag{2.2}$$

where π is equivalent to f. Generally speaking, the above technique is adopted for two purposes. Firstly, to evaluate the expectation of some function $h(Y)$, if Y is hard to sample from density f, one could alternatively rewrite the integral as an expectation of some function of X whose density π is easy to sample. In our example, it is easy to sample from $f(x) = \Phi_S(\mathbf{z})$. The second purpose is to reduce the variance. We want to choose π to make the estimator's variance as small as possible. Now consider the following density function

$$\pi(\mathbf{z}) = c\left(1 - \Phi\left(\frac{l - \mu_N(\mathbf{z})}{\sigma_N(\mathbf{z})}\right)\right)\phi(\mathbf{z}) = cg(\mathbf{z}) \tag{2.3}$$

where ϕ is the probability density function (pdf) of the multivariate Gaussian and c is a normalizing constant such that $\pi(\mathbf{z})$ is a proper pdf. It's easy to see from equation (1.13) that, if we are able to sample the $\mathcal{Z}$ from $\pi(\mathbf{z})$, then

$$\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\} = \mathbb{E}_\pi\left[\frac{\left(1 - \Phi\left(\frac{l-\mu_N(\mathcal{Z})}{\sigma_N(\mathcal{Z})}\right)\right)\phi(\mathcal{Z})}{\pi(\mathcal{Z})}\right] = \mathbb{E}_\pi\left[\frac{1}{c}\right] = \frac{1}{c} \tag{2.4}$$

Thus, finding the scaling constant c, that makes $\pi(\mathbf{z})$ a pdf is essentially as hard as approximating the integral on the right hand side of (2.1). However, by choosing such a density function, we have reduced

the variance to zero. Therefore we call $\pi(\mathbf{z})$ the *zero variance function* and our goal is to find an easy-to-sample density function $\pi^*(\mathbf{z})$ which approximates the zero variance density function, $\pi(\mathbf{z})$, and reduces the variance, i.e.

$$\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\} = \mathbb{E}_{\pi^*}\left[\frac{\left(1-\Phi\left(\frac{l-\mu_N(\mathcal{Z})}{\sigma_N(\mathcal{Z})}\right)\right)\phi(\mathcal{Z})}{\pi^*(\mathcal{Z})}\right], \quad \text{where } \pi^*(\mathbf{z}) \approx \pi(\mathbf{z}) \tag{2.5}$$

We give an indication of what this zero variance function looks like. From equation (2.3), $\pi(\mathbf{z})$ is a product of a Gaussian pdf and a Gaussian cdf. Figures 2.1 and 2.2 depict such a function generated from the configuration and settings in Chapter 5.

Figure 2.1 shows the shape of the zero variance function $\pi(\mathbf{z})$ in the univariate case. When the loss is small, as in Figure 2.1a, a wide range of systematic risk factors will give such a loss level, while this range shrinks as loss percentage increases. One interesting characteristic for the 1D model is that the shape of the zero variance function matches that of the Gaussian Mixture Model with two components. As the loss percentage increases, these two humps separate and move in different directions. At some level, one component disappears and the zero variance function looks like a univariate Gaussian.

When the risk factor is bivariate, the zero variance function is pictured in Figure 2.2. Unlike the univariate case where $\pi(\mathbf{z})$ could be modeled as a Gaussian Mixture with 2 components, the zero variance function forms a volcano shape. All three figures on the right side give a bird's eye view of the density. As l increases, the edges expand outwards slightly and decay at different speeds. In Figure 2.2f, the extreme value of l concentrates the risk factor plane to a tiny dark red region corresponding to a "black swan event". Most of the points on the risk factor plane do not cause huge losses. However, if this black swan event is triggered, the portfolio holder will suffer significant losses. Hence, naturally, sampling from $\pi(\mathbf{z})$ is equivalent, in a sense, to rare event simulation.

2.2 Cross Entropy

The cross-entropy (CE) method is an efficient and simple approach in rare-event simulation based on an iterative importance sampling procedure. The method uses the Kullback-Leibler (KL) distance (also known as cross entropy) that gives a "distance" between two probability measures. Consider the estimation of the probability

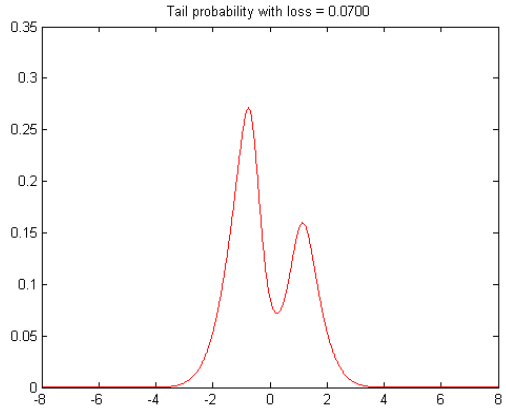
$$q = \mathbb{P}(S(\mathcal{X}) \geqslant l) = \mathbb{E}[\mathbb{1}_{\{S(\mathcal{X})\geqslant l\}}] = \int \mathbb{1}_{\{S(\mathbf{x})\geqslant l\}} f(\mathbf{x};\mathbf{u}) d\mathbf{x}$$

where S is a real-valued function, l is a threshold and $\mathcal{X}$ follows some probability density function $f(\cdot;\mathbf{u})$ parametrized by $\mathbf{u}$. Rather than sampling directly from f, one natural way we have seen already is to use importance sampling based on another pdf g such that

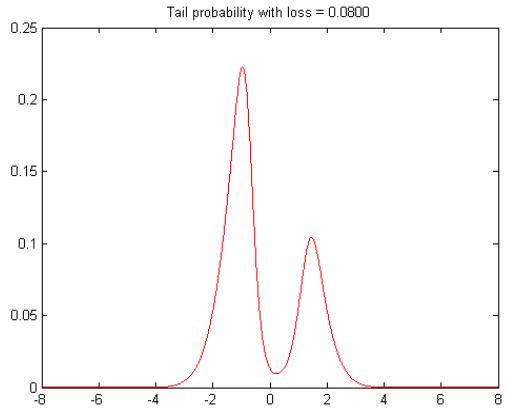
$$q = \int \frac{\mathbb{1}_{\{S(\mathbf{x})\geqslant l\}} f(\mathbf{x};\mathbf{u})}{g(\mathbf{x})} g(\mathbf{x}) d\mathbf{x} = \mathbb{E}\left[\frac{\mathbb{1}_{\{S(\mathcal{X})\geqslant l\}} f(\mathcal{X};\mathbf{u})}{g(\mathcal{X})}\right], \qquad \mathcal{X} \sim g.$$

Note that here the zero-variance function is $g^*(\mathbf{x}) = q^{-1}\mathbb{1}_{\{S(\mathbf{x})\geqslant l\}} f(\mathbf{x};\mathbf{u})$

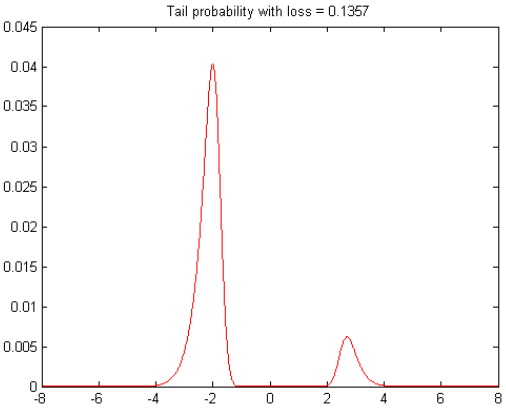
Let f and g be two probability distribution functions associated with a vector $\mathbf{x}$. The KL distance



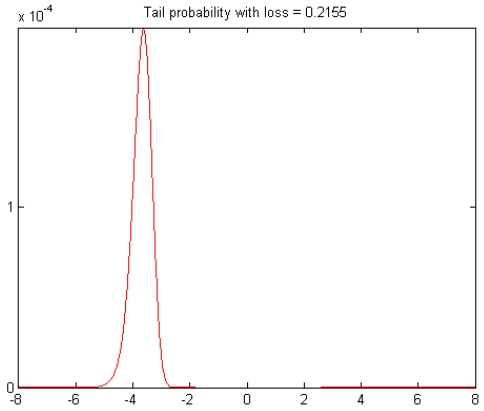
(a) zero variance for $l = 0.0700$



(b) zero variance for $l = 0.0800$



(c) zero variance for $l = 0.1357$



(d) zero variance for $l = 0.2155$

Figure 2.1: 1D zero variance function for various loss levels

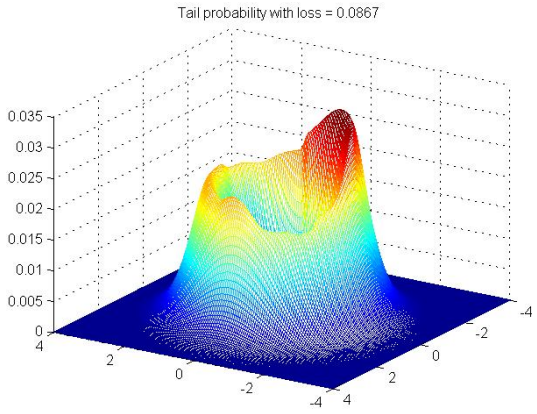
is defined as

$$\mathcal{D}(f, g) = \int f(\mathbf{x}) \log \frac{f(\mathbf{x})}{g(\mathbf{x})} d\mathbf{x}.$$

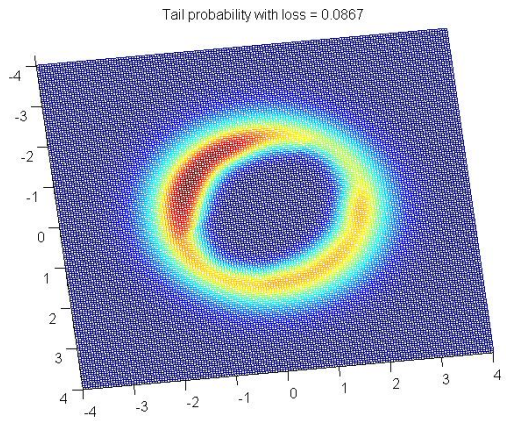
The CE method looks for a density function g such that $\mathcal{D}(g^*, g)$ is minimized from the parametric family $\mathcal{F} = \{f(\mathbf{x}; \mathbf{v})\}$. The optimal parameter vector $\mathbf{v}^*$ satisfies

$$\begin{aligned}
\mathbf{v}^* &= \arg\min_{\mathbf{v}} \mathcal{D}(g^*, f(\cdot; \mathbf{v})) \\
&= \arg\min_{\mathbf{v}} \left\{ \int g^*(\mathbf{x}) \log g^*(\mathbf{x}) d\mathbf{x} - q^{-1} \int f(\mathbf{x}; \mathbf{u}) \mathbb{1}_{\{S(\mathbf{x}) \geqslant l\}} \log f(\mathbf{x}; \mathbf{v}) d\mathbf{x} \right\} \\
&= \arg\max_{\mathbf{v}} \int f(\mathbf{x}; \mathbf{u}) \mathbb{1}_{\{S(\mathbf{x}) \geqslant l\}} \log f(\mathbf{x}; \mathbf{v}) d\mathbf{x} \qquad (2.6) \\
&= \arg\max_{\mathbf{v}} \int \mathbb{1}_{\{S(\mathbf{x}) \geqslant l\}} \log f(\mathbf{x}; \mathbf{v}) \frac{f(\mathbf{x}; \mathbf{u})}{f(\mathbf{x}; \mathbf{w})} f(\mathbf{x}; \mathbf{w}) d\mathbf{x} \qquad (2.7)
\end{aligned}$$

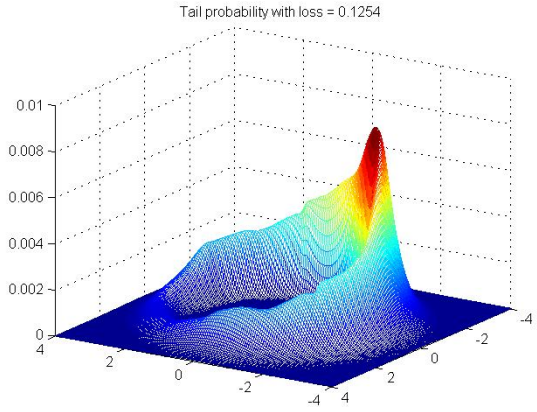
Several points are worth mentioning about the equations above. Instead of directly finding the optimal parameter vector $\mathbf{v}$ in equation (2.6), in practice, we would change the integral to a sum of Monte Carlo



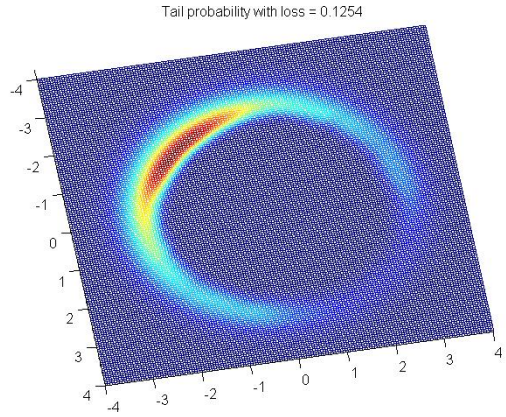
(a) zero variance for $l = 0.0867$



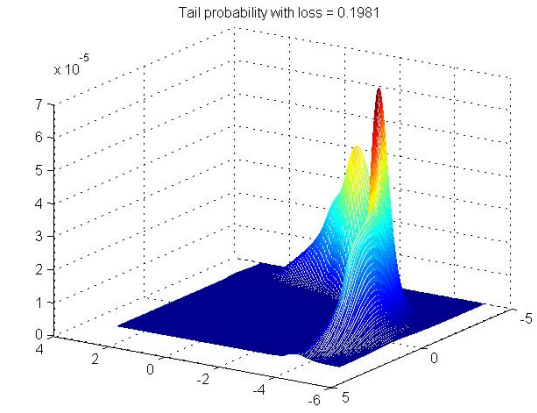
(b) zero variance for $l = 0.0867$



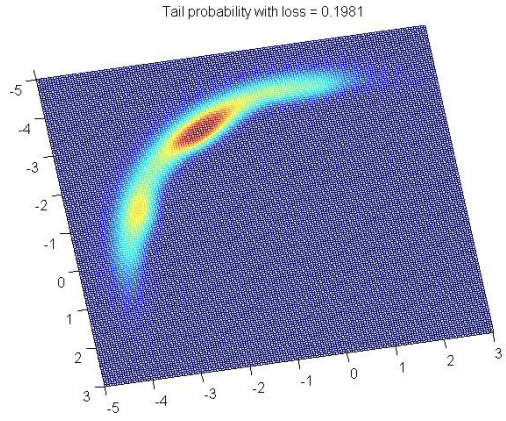
(c) zero variance for $l = 0.1254$



(d) zero variance for $l = 0.1254$



(e) zero variance for $l = 0.1981$



(f) zero variance for $l = 0.1981$

Figure 2.2: 2D zero variance function for various loss levels

samples as

$$\begin{aligned}\mathbf{v}^* &= \arg\max_{\mathbf{v}} \int f(\mathbf{x};\mathbf{u}) \mathbb{1}_{\{S(\mathbf{x})\geqslant l\}} \log f(\mathbf{x};\mathbf{v}) d\mathbf{x} \\ &\approx \arg\max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{S(\mathcal{X}_i)\geqslant l\}} \log f(\mathcal{X}_i;\mathbf{v}) \qquad \mathcal{X}_i \sim f(\cdot;\mathbf{u}) \end{aligned} \tag{2.8}$$

and then solve (2.8).

Moreover since $S(\mathcal{X}) \geqslant l$ is a rare event, most of the samples will result in $\mathbb{1}_{\{S(\mathcal{X}_i)\geqslant l\}}$ being 0. To overcome this problem, CE finds the optimal parameter $\mathbf{v}^*$ through a sequence $\mathbf{v}_0, \mathbf{v}_1, \ldots$ and in each iteration, an importance sampling technique is adopted and the threshold l is adaptively selected. More specifically, in the t^{th} iteration, $\mathbf{w}$ in the following discretization from equation (2.8) is set to be $\mathbf{v}_{t-1}$ to achieve importance sampling.

$$\begin{aligned}\mathbf{v}^* &= \arg\max_{\mathbf{v}} \int \mathbb{1}_{\{S(\mathbf{x})\geqslant l\}} \log f(\mathbf{x};\mathbf{v}) \frac{f(\mathbf{x};\mathbf{u})}{f(\mathbf{x};\mathbf{w})} f(\mathbf{x};\mathbf{w}) d\mathbf{x} \\ &\approx \arg\max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{S(\mathcal{X}_i)\geqslant l\}} \log f(\mathcal{X}_i;\mathbf{v}) \frac{f(\mathcal{X}_i;\mathbf{u})}{f(\mathcal{X}_i;\mathbf{w})} \qquad \mathcal{X}_i \sim f(\cdot;\mathbf{w}) \end{aligned} \tag{2.9}$$

For all sampled values, we then choose l_t from the top quantile so that we have enough non zero evaluations of the indicator function. Lastly $\mathbf{v}$ is maximized for the t^{th} iteration. We expect to see that the sequence $\mathbf{v}_0, \mathbf{v}_1, \ldots$ converges to $\mathbf{v}^*$ and $l_0, l_1, \ldots$ converges to l. This approach is outlined in the multi-level algorithm [16] below:

Algorithm 1 CE Algorithm for Rare-Event Estimation

Input: sample size N and the parameter $\rho \in (0,1)$

1: Set initial parameters $\mathbf{v}_0$. Let $N^e = \lceil \rho N \rceil$. Set $t = 1$
2: Generate $\mathcal{X}_1, \ldots, \mathcal{X}_N \sim_{iid} f(\cdot;\mathbf{v}_{t-1})$. Calculate $S_i = S(\mathcal{X}_i)$ for all i and order these from smallest to largest: $S_{(1)} \leqslant \cdots \leqslant S_{(N)}$. Let l_t be the sample $(1-\rho)$-quantile of performances; that is $l_t = S_{(N-N^e+1)}$. if $l_t > l$, reset l_t to l.
3: Use the same sample $\mathcal{X}_1, \ldots, \mathcal{X}_N$ to solve equation (2.9) with $\mathbf{w} = \mathbf{v}_{t-1}$. Denote the solution by $\mathbf{v}_t$.
4: If $l_t < l$, set $t = t + 1$ and reiterate from Step 2; otherwise proceed with step 5.
5: let $T = t$ be the final iteration counter. Generate $\mathcal{X}_1, \ldots, \mathcal{X}_N \sim_{iid} f(\cdot;\mathbf{v}_T)$ and estimate q via importance sampling.

To apply the CE approach to our problem, recall the zero variance function $\pi(\mathbf{z}) = cg(\mathbf{z})$ in equation (2.3). Hence our goal is to look for $\pi^*(\mathbf{z})$ such that the KL distance $\mathcal{D}(cg(\mathbf{z}), \pi^*(\mathbf{z}))$ is minimized. Further to simplify the log optimization in equation (2.9), we assume $\pi^*(\mathbf{z})$ belongs to the natural exponential family. More specifically, $\pi^*(\mathbf{z}) = \phi(\mathbf{z};\mu,\Sigma)$ where ϕ is the multivariate Gaussian pdf. As a result, $\mathbf{v} = \{\mu, \Sigma\}$ and

$$\begin{aligned}\mathbf{v}_t &= \arg\min_{\mathbf{v}} \mathcal{D}(cg(\mathbf{z}), \phi(\mathbf{z};\mathbf{v})) \\ &\approx \arg\max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} g(\mathbf{X}_i) \log \phi(\mathbf{X}_i;\mathbf{v}) \\ &\approx \arg\max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} \frac{g(\mathbf{X}_i)}{\phi(\mathbf{X}_i;\mathbf{v}_{t-1})} \log \phi(\mathbf{X}_i;\mathbf{v}) \qquad \mathbf{X}_i \sim \phi(\cdot;\mathbf{v}_{t-1}) \end{aligned} \tag{2.10}$$

It's worth pointing out that, during the iteration, $\mathbf{v}_t$ may be trapped in local optima. Rubinstein and Kroese [17] propose the use of dynamic smoothing to overcome this issue. On the other hand, Chan and Kroese point out that, for high dimensional problems, this multi-level CE approach becomes unstable and may not give a desirable solution [2].

One-level CE

So far, we have assumed that l is a threshold such that $S(\mathcal{X}) \geqslant l$ is a rare event. Recall that the zero-variance function is $g^*(\mathbf{x}) = q^{-1}\mathbb{1}_{\{S(\mathbf{x})\geqslant l\}}f(\mathbf{x};\mathbf{u})$ and it is hard to sample from this function. However, if random samples $\mathcal{X}_1,\ldots,\mathcal{X}_N$ could be directly drawn from $g^*(\mathbf{x})$, we could reduce the multi-level cross entropy method to a one-level CE method, as equation (2.9) becomes

$$\begin{aligned}\mathbf{v}^* &\approx \arg\max_{\mathbf{v}} \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{\{S(\mathcal{X}_i)\geqslant l\}} \log f(\mathcal{X}_i;\mathbf{v}) \frac{f(\mathcal{X}_i;\mathbf{u})}{f(\mathcal{X}_i;\mathbf{w})} \qquad \mathcal{X}_i \sim f(\cdot;\mathbf{w}) \\ &\approx \arg\max_{\mathbf{v}} \frac{1}{N}\sum_{i=1}^{N} \log f(\mathcal{X}_i;\mathbf{v}) \qquad \mathcal{X}_i \sim g^* \end{aligned} \tag{2.11}$$

As a result, in this case, equation (2.11) is nothing but maximizing the log likelihood. The improvement is two folds. On the one hand, the CE instability is overcome and, on the other hand, more importantly, we approximately solve the optimization problem in one iteration only, which is much faster than the standard CE algorithm.

Loosely speaking, in our problem, if the iterative CE algorithm works, $\phi(\cdot;\mathbf{v}_t)$ must be close to the target zero variance function $\pi(\mathbf{z}) = cg(\mathbf{z})$ as $t \to \infty$. Furthermore, since $\pi(\mathbf{z}) = cg(\mathbf{z})$, where $g(\mathbf{z})$ is a product of a cdf and a pdf, we can reduce the complexity from a multi-level CE algorithm to a one-level CE algorithm by directly drawing samples $\mathcal{X}_1,\ldots,\mathcal{X}_N$ from $\pi(\mathbf{z})$ through Markov Chain Monte Carlo. Equation (2.10) hence becomes

$$\mathbf{v}^* \approx \arg\max_{\mathbf{v}} \frac{1}{N}\sum_{i=1}^{N} \log \phi(\mathcal{X}_i;\mathbf{v}) \qquad \mathcal{X}_i \sim \pi \tag{2.12}$$

Note that $\phi(\cdot;\mathbf{v}_t)$ is no longer limited to the exponential family, unless the argument optimization is too difficult to solve. In the following subsections, we first outline Markov Chain Monte Carlo, followed by a review of the Gaussian Mixture Model.

2.3 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a sampling method based on constructing a Markov Chain whose equilibrium distribution is the probability distribution we want to sample from. In other words, given a complicated density $\pi = cg$, where c is a constant, we aim to construct a Markov chain $X_0, X_1, \ldots$ such that $X_n \sim \pi$ for large n. A key point for a good MCMC is the proposal move, which is the transition from state X_{n-1} to state X_n. We start with a review of the random walk proposal, namely the Metropolis Sampling Algorithm. Then some other techniques are investigated.

2.3.1 Metropolis Sampler

Let x and y be two states in the state space E. Let $q(x,y)$ be the probability of transiting from state x to state y. This $q(x,y)$ is also called the proposal distribution at state x. Further define the acceptance

probability $\alpha(x, y)$ to be

$$\alpha(x,y) = \begin{cases} \min\left\{\frac{\pi(y)}{\pi(x)}\frac{q(y,x)}{q(x,y)}, 1\right\} & \text{if } \pi(x)q(x,y) \neq 0 \\ 1 & \text{if } \pi(x)q(x,y) = 0 \end{cases} \tag{2.13}$$

The probability for a successful jump from state x to state y is $q(x, y)\alpha(x, y)$. A chain is reversible, if $\forall x, y \in E$, $\pi(x)q(x, y)\alpha(x, y) = \pi(y)q(y, x)\alpha(y, x)$. One can show that, with $\alpha(x, y)$ defined in equation (2.13), the chain is reversible. Further, if $q(x, y)$ is chosen to be multivariate normal, where $y \sim MVN(x, \sigma^2\mathbf{I})$, then it is obvious that $q(x, y) = q(y, x)$. This symmetry simplifies $\alpha(x, y)$ to

$$\alpha(x,y) = \begin{cases} \min\left\{\frac{\pi(y)}{\pi(x)}, 1\right\} & \text{if } \pi(x)q(x,y) \neq 0 \\ 1 & \text{if } \pi(x)q(x,y) = 0 \end{cases} \tag{2.14}$$

The Metropolis-Hasting Sampling based on random walks and symmetric proposal moves is presented in algorithm 2.

Algorithm 2 Metropolis-Hasting Algorithm

1: Choose some initial value X_0
2: **for** $n = 1, \ldots, M$ **do**
3: choose a proposal move $Y \sim MVN(X_{n-1}, \sigma^2\mathbf{I})$
4: Let $A = \pi(Y)/\pi(X_{n-1}) = g(Y)/g(X_{n-1})$, and $U_n \sim$ Uniform$[0, 1]$
5: **if** $U_n < A$ **then**
6: accept the proposal move by setting $X_n = Y$
7: **else**
8: reject the proposal by setting $X_n = X_{n-1}$
9: **end if**
10: **end for**
11: **return** $X_{B+1}, \ldots, X_M$ where B ("burn-in") is large enough that $X_B \sim \pi$ (approximately).

There are a few criteria used to evaluate the performance of a MCMC algorithm. One intuitive standard is the acceptance rate. Generally speaking, if σ is large in the proposal distribution $MVN(X_{n-1}, \sigma^2\mathbf{I})$, then the proposed sample is usually rejected. This makes the chain stuck at one state and it does not move much. On the other hand, if σ is too small, proposed samples are close to the current state and this results in acceptance most of the time. This is not desirable either, since the whole space is not sufficiently explored. Hence scaling (i.e., σ) must be chosen carefully and the acceptance rate plays a key role in optimization. Roberts et al. [1] and Roberts and Rosenthal [14] show that in a certain idealized high-dimensional limit, the optimal acceptance rate is 0.234. Therefore we can adapt the MCMC method by tuning σ along the chain, in order for the acceptance rate to approximate this magic number.

2.3.2 Variable-At-A-Time and Parallel Tempering MCMC

The Metropolis sampler provides an effective way to sample from a complicated density. Steps in the algorithm are intuitive and easy to implement. One goal for a good proposal distribution is to make the sample X_n have a density approximately equal to π in relatively few steps. The naive MCMC method described in Algorithm 2 takes a long time to explore the whole space, especially if the regime is a high-dimensional space or the density π has some "traps". In this section, we review two MCMC methods

which overcome these issues.

Variable-At-A-Time MCMC When states are multidimensional ($X_n \in \mathcal{R}^d$), it is often hard for the sampler to move a step if all dimensions are changed simultaneously. Consequently rather than changing all coordinates at once, the sampler could make a step by changing one coordinate at a time while holding all the other coordinates unchanged. The changing coordinate can be selected randomly from $1, \ldots, d$ or each component can be changed in order. Algorithm 3 gives an outline for this Variable-At-A-Time MCMC, where $X_{n-1,j}$ stands for the j^{th} component in the sample X_{n-1} and $X_{n-1,-j}$ stands for all the components in the sample X_{n-1} with the j^{th} component excluded.

Algorithm 3 Variable-At-A-Time MCMC

Choose some initial value X_0
for $n = 1, \ldots, N$ **do**
  for $j = 1, \ldots, d$ **do**
    Let $Y_{-j} = X_{n-1,-j}$ and choose a proposal move $Y_j \sim N(X_{n-1,j}, \sigma_j^2)$
    Let $A = \pi(Y)/\pi(X_{n-1}) = g(Y)/g(X_{n-1})$, and $U_n \sim$ Uniform$[0, 1]$
    if $U_n < A$ **then**
      accept the proposal move by setting $X_n = Y$
    else
      reject the proposal by setting $X_n = X_{n-1}$
    end if
  end for
end for
return $X_{B+1}, \ldots, X_N$ where B ("burn-in") is large enough that $X_B \sim \pi$ (approximately).

Parallel Tempering MCMC As we have seen in section 2.1, the zero variance function is bimodal in the univariate risk factor space. In the multivariate case, the 2D example in Figure 2.2f gives two separated regions, where the dark red one is the more significant region while the yellow one is a decayed portion. Such bimodal distributions often trap plain MCMC near one mode and it fails to explore the whole space.

One observation for the sampling distribution is that the flatter the density is, the easier for the MCMC sampler to travel. For example, $\pi_1 = \frac{1}{2}N(0, 1) + \frac{1}{2}N(20, 1)$ and $\pi_2 = \frac{1}{2}N(0, 10^2) + \frac{1}{2}N(20, 10^2)$ each have two significant humps far away from each other. However, π_2 is flatter and easier for plain MCMC to handle than π_1. Parallel Tempering MCMC aims to overcome the problem of being trapped by introducing a few paralleled Markov chains, each of which is associated with a different "temperature". The "coldest" temperature is associated with the original density; increasing the temperature makes the distribution flatter. The key idea is to have the chain with highest temperature explore the whole space. The coldest one, on the other hand, explores some hump for a few steps and gets driven to other humps by the "hottest" parallel chains.

A common trick to create a density at a high temperature τ is to set $\pi_\tau(x) = c_\tau(\pi(x))^{1/\tau}$, where c_τ is a normalizing constant. Instead of treating each chain separately, parallel tempering MCMC with m chains has the stationary distribution $\Pi = \pi_1 \times \pi_2 \times \cdots \times \pi_m$ as a joint density. Now sampling can be broken down into two step. Within each chain, a new value $Y_{n,\tau}$ is proposed from some symmetric distribution, such as $Y_{n,\tau} \sim N(X_{n-1,\tau}, \sigma_\tau \mathbf{I})$. This new value is accepted with the probability $\min\left(1, \frac{\pi_\tau(Y_{n,\tau})}{\pi_\tau(X_{n-1,\tau})}\right)$. There is a random swap among two chains τ and τ' with the acceptance probability $\min\left(1, \frac{\pi_\tau(X_{n,\tau'})\pi_{\tau'}(X_{n,\tau})}{\pi_\tau(X_{n,\tau})\pi_{\tau'}(X_{n,\tau'})}\right)$,

which further simplifies to

$$\min\left(1, \frac{c_\tau \pi(X_{n,\tau'})^{1/\tau} c_{\tau'} \pi(X_{n,\tau})^{1/\tau'}}{c_\tau \pi(X_{n,\tau})^{1/\tau} c_{\tau'} \pi(X_{n,\tau'})^{1/\tau'}}\right) = \min\left(1, \frac{\pi(X_{n,\tau'})^{1/\tau} \pi(X_{n,\tau})^{1/\tau'}}{\pi(X_{n,\tau})^{1/\tau} \pi(X_{n,\tau'})^{1/\tau'}}\right) \tag{2.15}$$

The "coldest" chain gives the samples with target distribution π. Algorithm 4 gives the Parallel Tempering MCMC in detail. For a discussion of the ergodicity and correctness of this algorithm, see Neal [9] and Rosenthal [15].

Algorithm 4 Parallel Tempering MCMC

Choose some initial value $X_{0,j}$ for the j^{th} chain for $j = 1, 2, \ldots, m$.
for $n = 1, \ldots, N$ **do**
  for $j = 1, \ldots, m$ **do**
    Choose a proposal move $Y \sim N(X_{n-1,j}, \sigma_j^2 \mathbf{I})$
    Let $A = \pi_j(Y)/\pi_j(X_{n-1}) = (\pi(Y)/\pi(X_{n-1,j}))^{1/j}$, and $U_n \sim$ Uniform$[0, 1]$
    if $U_n < A_n$ **then**
      accept the proposal move by setting $X_{n,j} = Y$
    else
      reject the proposal by setting $X_{n,j} = X_{n-1,j}$
    end if
  end for
  Randomly choose τ and τ' from $1, \ldots, m$
  Switch $X_{n,\tau}$ and $X_{n,\tau'}$ with probability defined in equation (2.15)
end for
return $X_{B+1}, \ldots, X_N$ where B ("burn-in") is large enough that $X_B \sim \pi$ (approximately).

2.3.3 Slice Sampling

Both methods in the previous section are often better samplers than the simple Random Walk Metropolis. Variable-At-A-Time makes a movement easier in high dimensions while Parallel Tempering can sample more effectively from a multimodal distribution. Nevertheless, these methods share a common potential inefficiency: an appropriate "proposal" distribution is required. Enlightened by sampling uniformly from the region beneath the pdf curve in a univariate distribution, Neal [10] introduced a new Markov Chain Monte Carlo method called slice sampling, which avoids the proposal distribution and can adaptively change the scale size. We now provide a brief review of this sampling method.

Firstly slice sampling is based on auxilary sampling. Suppose $\pi(x) = cg(x)$ and one aims to sample from $\pi(x)$. Note that, if $(X, Y) \sim$ Uniform$\{(x, y) \in \mathbb{R}^2 : 0 \le y \le g(x)\}$, i.e., (X, Y) is chosen uniformly from the region bounded by g, then $X \sim \pi$. This is because the joint density for (X, Y) is

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

where $Z = \int g(x)dx = 1/c$. The marginal density for x is then

$$p(x) = \int_0^{g(x)} (1/Z)dy = g(x)/Z = cg(x) = \pi(x) \tag{2.16}$$

Hence, to sample for x, one can sample jointly for (x, y) and then ignore y. However sampling directly

from the region below g is hard. Neal proposed a two-stage sampling technique. Firstly, given the current x, y is sampled uniformly from the interval $(0, g(x))$. Then given the newly generated y, x is sampled over the region $S = \{x : y < g(x)\}$, which is a slice defined by y. The algorithm proceeds by alternatively sampling x and y.

One difficulty that arises in slice sampling is how to sample over the region $S = \{x : y < f(x)\}$. In the univariate case, one solution is to expand S to an interval $[a, b]$ that contains S. Then, in the sampling phase, sample x from $[a, b]$, but reject x and try again if $x \notin S$. In the multivariate case, one idea is to replace the interval in the univariate case by a hyperrectangle, where each dimension of the hyperrectangle contains the slice in that dimension. More advanced slice sampling approaches are discussed in [10].

2.4 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a linear combination of regular Gaussians. Generally speaking, a non-Bayesian GMM probability density function is of the form

$$\sum_{k=1}^{K} \alpha_k \phi(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{subject to} \qquad \sum_{k=1}^{K} \alpha_k = 1, \qquad \alpha_k \geq 0 \tag{2.17}$$

where K is the number of components. In this research project, GMM is selected to model the sampling data mainly for three reasons. Firstly, a GMM is easy to sample. Given all parameters, the sampling proceeds in two stages. Firstly i is sampled from a multinomial distribution $MULTNOM(\alpha_1, \ldots, \alpha_K)$. Then $Z \sim \phi(0, \mathbf{I})$ is sampled and $X = \boldsymbol{\mu}_i + \boldsymbol{L}_i Z$, where $\boldsymbol{L}_i$ is the Cholesky decomposition of $\boldsymbol{\Sigma}_i$. This X follows the GMM distribution (2.17). The second reason why GMM is considered comes from section 2.1, as the multi-modal shape of the risk-factor surface suggests GMM is a good candidate. Last, but not least, the Expectation-Maximization clustering algorithm performs quite well for a GMM. Our zero variance function can be approximated by a GMM, whose coefficients are determined by an optimization process that uses the training samples generated by the Markov Chain Monte Carlo method we reviewed in the previous section.

2.4.1 Expectation-Maximization

Expectation-Maximization (EM) is a common learning algorithm for Gaussian Mixture Models. It can also be applied to many other parametric models. Here we briefly review a generalization of EM. Suppose x is the observed data, $\boldsymbol{\theta}$ is the model parameter and z is a latent variable. Our aim is to find the $\boldsymbol{\theta}$ that maximizes the likelihood $P(x|\boldsymbol{\theta}) = \int P(x, z|\boldsymbol{\theta}) dz$. Instead of solving this optimization directly, the EM algorithm starts by maximizing a lower bound of the log likelihood:

$$\log \int P(x, z|\boldsymbol{\theta}) dz = \log \int \frac{P(x, z|\boldsymbol{\theta})}{Q(z)} Q(z) dz = \log \mathbb{E}_Q \left[\frac{P(x, z|\boldsymbol{\theta})}{Q(z)} \right] \geqslant \mathbb{E}_Q \left[\log \frac{P(x, z|\boldsymbol{\theta})}{Q(z)} \right] \tag{2.18}$$

where the last inequality in (2.18) follows from the Jensen's inequality.

Now let the function F of Q and θ be the last term in equation (2.18):

$$\begin{aligned} F(Q,\boldsymbol{\theta}) &= \mathbb{E}_Q\left[\log\frac{P(x,z|\boldsymbol{\theta})}{Q(z)}\right] \\ &= \mathbb{E}_Q[\log P(x,z|\boldsymbol{\theta})] - \mathbb{E}_Q[\log Q(z)] \qquad (2.19) \\ &= \log P(x|\theta) + \mathbb{E}_Q[\log P(z|x,\boldsymbol{\theta})] - \mathbb{E}_Q[\log Q(z)] \\ &= \log P(x|\boldsymbol{\theta}) - \mathbb{E}_Q[\log(Q(z)/P(z|x,\boldsymbol{\theta}))] \qquad (2.20) \end{aligned}$$

The last term in equation (2.20) is the KL distance between $Q(z)$ and $P(z|x,\boldsymbol{\theta})$. In fact, we will see below that we can set $Q(z) = P(z|x,\boldsymbol{\theta})$, making the KL distance equal to 0. The generalization of EM is:

- **E Step** - Use the current value of $\boldsymbol{\theta}$ and the observations of x to find the distribution Q for the latent variable z, i.e., $Q(z) = P(z|x,\boldsymbol{\theta})$

- **M step** - Maximize the expected value of $\log P(x,z|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, using the distribution Q found in the E step, i.e., $\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_Q[\log P(x,z|\boldsymbol{\theta})]$

Intuitively, the EM algorithm uses a two step process to maximize the lower bound $\mathbb{E}_Q\left[\log\frac{P(x,z|\boldsymbol{\theta})}{Q(z)}\right] = F(Q,\boldsymbol{\theta})$ of (2.18). The E step maximizes $F(Q,\boldsymbol{\theta})$, with respect to Q, by minimizing the KL distance between $Q(z)$ and $P(z|x,\boldsymbol{\theta})$ (i.e., the last term in (2.20)) by setting $Q(z) = P(z|x,\boldsymbol{\theta})$. The M step maximizes $F(Q,\boldsymbol{\theta})$, with respect to $\boldsymbol{\theta}$, by finding $\boldsymbol{\theta}$ to maximize $\mathbb{E}_Q[\log P(x,z|\boldsymbol{\theta})]$ (i.e., the first term in ((2.19))).

EM for Gaussian Mixture Models Specifically for d-dimensional Gaussian Mixture Models, $\boldsymbol{\theta} = \{\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1,\ldots,K}$ and the latent variable $z_{ik} = 1$ if the i^{th} observation comes from the k^{th} component and 0 otherwise. Following the generalized framework above, in the E step, we find the distribution Q of z_{ik} given the i^{th} observation and the parameters. By applying Bayes' Rule,

$$r_{ik} = P(z_{ik} = 1|x_i,\boldsymbol{\theta}) = \frac{\alpha_k\phi(X_i|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{k'=1}^{K}\alpha_{k'}\phi(X_i|\boldsymbol{\mu}_{k'},\boldsymbol{\Sigma}_{k'})} \qquad (2.21)$$

for $i = 1,\ldots,M$ and $k = 1,\ldots,K$, where r_{ik} is understood as the "responsibility" of component k, $X_i \sim \Pi$, $i = 1,\ldots,M$, generated by MCMC (the slice sampler, in our case), and $\phi(X_i|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$ is a d-dimension normal pdf with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$. Since we know $P(z|x,\boldsymbol{\theta})$, we can set $Q(z) = P(z|x,\boldsymbol{\theta})$

Since the Gaussian distribution belongs to the exponential family, we can maximize the expected value of the log likelihood in step M easily. In fact, for the i^{th} observation, the log probability is given as

$$\log\left[\prod_{k=1}^{K}\left(\alpha_k(2\pi)^{-\frac{d}{2}}|\boldsymbol{\Sigma}_k|^{-\frac{1}{2}}\exp(-\frac{1}{2}(X_i-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(X_i-\boldsymbol{\mu}_k)\right)^{z_{ik}}\right] \qquad (2.22)$$

Further taking expectation under the measure Q, i.e., the distribution of $z_{i,k}$, we have

$$\begin{aligned}
&\mathbb{E}_Q\left\{\log\left[\prod_{k=1}^{K}\left(\alpha_k(2\pi)^{-\frac{d}{2}}|\mathbf{\Sigma}_k|^{-\frac{1}{2}}\exp(-\frac{1}{2}(X_i-\boldsymbol{\mu}_k)^T\mathbf{\Sigma}_k^{-1}(X_i-\boldsymbol{\mu}_k)\right)^{z_{ik}}\right]\right\}\\
=\ &\mathbb{E}_Q\left\{\sum_{k=1}^{K}z_{ik}\left(\log(\alpha_k)-\frac{d}{2}\log(2\pi)-\frac{1}{2}\log(|\mathbf{\Sigma}_k|)-\frac{1}{2}(X_i-\boldsymbol{\mu}_k)^T\mathbf{\Sigma}_k^{-1}(X_i-\boldsymbol{\mu}_k)\right)\right\}\\
=\ &\sum_{k=1}^{K}r_{ik}\left(\log(\alpha_k)-\frac{d}{2}\log(2\pi)-\frac{1}{2}\log(|\mathbf{\Sigma}_k|)-\frac{1}{2}(X_i-\boldsymbol{\mu}_k)^T\mathbf{\Sigma}_k^{-1}(X_i-\boldsymbol{\mu}_k)\right) \qquad (2.23)
\end{aligned}$$

where $r_{ik}=\mathbb{E}_Q(z_{ik})$. To maximize the sum (2.23) for all M samples, we simply maximize

$$\sum_{i=1}^{M}\sum_{k=1}^{K}r_{ik}\log(\alpha_k)$$

with respect to α,

$$-\frac{1}{2}\sum_{i=1}^{M}\sum_{k=1}^{K}r_{ik}(X_i-\boldsymbol{\mu}_k)^T\mathbf{\Sigma}_k^{-1}(X_i-\boldsymbol{\mu}_k)$$

with respect to $\boldsymbol{\mu}_k$, and

$$-\frac{1}{2}\sum_{i=1}^{M}\sum_{k=1}^{K}r_{ik}\left(\log|\mathbf{\Sigma}_k|+(X_i-\boldsymbol{\mu}_k)^T\mathbf{\Sigma}_k^{-1}(X_i-\boldsymbol{\mu}_k)\right)$$

with respect to $\mathbf{\Sigma}_k$. After some algebraic simplification, the M step can be described as the parameter re-estimation, with weights r_{ik} given from the E step:

$$\alpha_k=\frac{1}{M}\sum_{i=1}^{M}r_{ik},\qquad \boldsymbol{\mu}_k=\frac{\sum_{i=1}^{M}r_{ik}X_i}{\sum_{i=1}^{M}r_{ik}},\qquad \mathbf{\Sigma}_k^2=\frac{\sum_{i=1}^{M}r_{ik}(X_i-\boldsymbol{\mu}_k)^T(X_i-\boldsymbol{\mu}_k)}{\sum_{i=1}^{M}r_{ik}} \qquad (2.24)$$

Started with some random guess for the parameter $\boldsymbol{\theta}_0$, the above two steps are iterated until $\boldsymbol{\theta}$ converges.

2.4.2 Drawbacks of EM

EM is a straightforward algorithm for data clustering and, for a Gaussian Mixture Model, it is simple to implement. However a few drawbacks are worth mentioning.

Firstly, EM requires a predetermined number of Gaussian components. As shown in Figure 2.1, we can assume two components are enough in the univariate case to characterize the zero variance function. However, for higher dimensions, although the ring shaped density in Figure 2.2 can still be approximated by a Gaussian Mixture Model, it is not obvious how to choose the number of Gaussian components needed for a good approximation. Moreover, over-fitting happens if too many components are selected.

Secondly, both the convergence speed and the quality of the approximation may be issues. Consider the case where one component of a Gaussian Mixture Model has mean $\boldsymbol{\mu}_k$ equal to some data observation X_i and covariance matrix $\mathbf{0}$. If the component weight α_k is not 0, this will give an infinite value and lead to a global maximum. That's to say, we want EM to converge to some local optimum, rather than a global optimum. This singularity of the covariance matrix should be avoided, but can't be guaranteed.

Chapter 3

Bayesian Monte Carlo

3.1 Bayesian Linear Model

A Gaussian Process is used to define distributions over functions. It derives from a Bayesian linear basis function model in which the number of basis functions goes to infinity. Before looking at the Gaussian Process Model, we begin with a basic model which has only a finite number of basis functions:

$$y = \phi(x)^T\beta + \epsilon, \qquad \epsilon \sim N(0, \sigma^2)$$

where $\phi(x)$ is a column vector of basis functions valued at a training case x, $\beta = [b_1, \cdots, b_n]^T$ is the vector of regression coefficients and ϵ is Gaussian noise with mean 0 and variance σ^2.

With n training cases in the set and observations $\mathbf{y} = [y_1, \ldots, y_n]^T$, we get that

$$\mathbf{y} \sim N(\Phi^T\beta, \sigma^2 I)$$

where $\Phi = [\phi_1, \cdots, \phi_n]$ with $\phi_k = \phi(x_k)$. Assume that the prior probability for β is $N(0, S_0)$. Then the prior probability for $\mathbf{y}$ will also be Gaussian because $\mathbf{y}$ is a linear function of β. From Bayes' rule,

$$\begin{aligned} p(\beta|\Phi, \mathbf{y}) &= \frac{p(\mathbf{y}|\Phi, \beta)p(\beta)}{p(\mathbf{y}|\Phi)} \\ &\propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \Phi^T\beta)^T(\mathbf{y} - \Phi^T\beta)\right)\exp\left(-\frac{1}{2}\beta^T S_0^{-1}\beta\right) \\ &\propto \exp\left(-\frac{1}{2}(\sigma^{-2}\mathbf{y}^T\mathbf{y} + \beta^T\Sigma\beta - \sigma^{-2}\mathbf{y}^T\Phi^T\beta - \sigma^{-2}\beta^T\Phi\mathbf{y})\right) \end{aligned}$$

where $\Sigma = \sigma^{-2}\Phi\Phi^T + S_0^{-1}$ is symmetric. In fact, we can do more algebraic simplification and find that

$$\begin{aligned} &\sigma^{-2}\mathbf{y}^T\mathbf{y} + \beta^T\Sigma\beta - \sigma^{-2}(\mathbf{y}^T\Phi^T\beta + \beta^T\Phi\mathbf{y}) \\ =&\beta^T\Sigma\beta - \sigma^{-2}(\mathbf{y}^T\Phi^T\Sigma^{-1}\Sigma\beta + \beta^T\Sigma\Sigma^{-1}\Phi\mathbf{y}) + \sigma^{-2}\mathbf{y}^T\mathbf{y} \\ =&(\beta - \bar{\beta})^T\Sigma(\beta - \bar{\beta}) + (\sigma^{-2}\mathbf{y}^T\mathbf{y} - \bar{\beta}^T\Sigma\bar{\beta}) \end{aligned}$$

where $\bar{\beta} = \sigma^{-2}\Sigma^{-1}\Phi\mathbf{y}$ and the last term is some constant. Thus, $p(\beta|\Phi, \mathbf{y}) \propto \exp\left(-\frac{1}{2}(\beta - \bar{\beta})^T\Sigma(\beta - \bar{\beta})\right)$, i.e., the posterior probability of β is Gaussian with mean $\bar{\beta}$ and covariance matrix Σ^{-1}, where $\Sigma =$

$\sigma^{-2}\Phi\Phi^T + S_0^{-1}$:

$$p(\beta|\Phi, \mathbf{y}) \sim N(\frac{1}{\sigma^2}\Sigma^{-1}\Phi\mathbf{y}, \Sigma^{-1}).$$

Now given a test case x_*, denote the vector of basis functions valued at x_* by $\phi_* = \phi(x_*)$. Our purpose is to infer y_* from observations $\mathbf{y}$ without reference to β. One approach is to marginalize β by computing the posterior probability of β and probability of y_* conditioned on β. This is called the posterior predictive distribution:

$$p(y_*|x_*, \Phi, \mathbf{y}) = \int p(y_*|x_*, \beta)p(\beta|\Phi, \mathbf{y})d\beta \sim N\left(\frac{1}{\sigma^2}\phi_*^T\Sigma^{-1}\Phi\mathbf{y}, \phi_*^T\Sigma^{-1}\phi_*\right) \tag{3.1}$$

Notice that both $p(y_*|x_*, \beta)$ and $p(\beta|\Phi, \mathbf{y})$ are Gaussians and the mean of y_* is affine in β. Hence the posterior predictive distribution is also Gaussian [12]. We use this technique again later.

Consider now the right side of equation (3.1). First note that the covariance matrix of the observations in the training set is $C = \sigma^2 I + \Phi^T S_0 \Phi$. The covariance of y_* itself is $v = \phi_*^T S_0 \phi_*$. Similarly the covariance of y_* and $\mathbf{y}$ is $k = \Phi^T S_0 \phi_*$. It is straightforward to show that

$$\begin{aligned}
k^T C^{-1}\mathbf{y} &= \phi_*^T S_0\Phi(\sigma^2 I + \Phi^T S_0\Phi)^{-1}\mathbf{y} \\
&= \frac{1}{\sigma^2}\phi_*^T(S_0\Phi(I + \frac{1}{\sigma^2}\Phi^T S_0\Phi)^{-1}\Phi^{-1})\Phi\mathbf{y} \\
&= \frac{1}{\sigma^2}\phi_*^T(S_0^{-1} + \sigma^{-2}\Phi\Phi^T)^{-1}\Phi\mathbf{y} \\
&= \frac{1}{\sigma^2}\phi_*^T\Sigma^{-1}\Phi\mathbf{y}
\end{aligned}$$

and

$$\begin{aligned}
v - k^T C^{-1}k &= \phi_*^T S_0\phi_* - \phi_*^T S_0\Phi(\sigma^2 I + \Phi^T S_0\Phi)^{-1}\Phi^T S_0\phi_* \\
&= \phi_*^T S_0\phi_* - \phi_*^T S_0(S_0 + \sigma^2\Phi^{-T}\Phi^{-1})^{-1}S_0\phi_* \\
&= \phi_*^T[S_0 - (S_0^{-1} + \sigma^2 S_0^{-1}\Phi^{-T}\Phi^{-1}S_0^{-1})^{-1}]\phi_* \\
&= \phi_*^T[S_0(\sigma^2 S_0^{-1}\Phi^{-T}\Phi^{-1}S_0^{-1})(S_0^{-1} + \sigma^2 S_0^{-1}\Phi^{-T}\Phi^{-1}S_0^{-1})^{-1}]\phi_* \\
&= \phi_*^T(S_0^{-1} + \sigma^{-2}\Phi\Phi^T)^{-1}\phi_* \\
&= \phi_*^T\Sigma^{-1}\phi_*
\end{aligned}$$

This shows that the posterior Gaussian can be fully characterized by the mean and variance given above, i.e.,

$$\mathbb{E}(y_*|\mathbf{y}) = k^T C^{-1}\mathbf{y}, \qquad \mathbb{V}(y_*|\mathbf{y}) = v - k^T C^{-1}k. \tag{3.2}$$

Both the mean and variance have an intuitive explanation. The mean, y_*, should be implied from observations $\mathbf{y}$ and correlations between y_* and $\mathbf{y}$. Given observations $\mathbf{y}$, one is able to reduce the uncertainty from v to $v - k^T C^{-1}k$. More specifically $k^T C^{-1}k$ is the information gained through training cases.

3.2 Gaussian Process Model

A Gaussian Process (GP) is a generalization of a Gaussian distribution. A Gaussian distribution, whose mean is a vector and covariance is a matrix, is used to describe probabilities over a vector. A Gaussian Process, on the other hand, works on functions instead. This process is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution [12]. For a random function $f(x)$, define its mean function $m(x)$ and its variance function $k(x, x')$ to be

$$m(x) = \mathbb{E}[f(x)], \qquad k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]. \tag{3.3}$$

Then $f(x)$ is modeled as a GP with mean function $m(x)$ and covariance function $k(x, x')$ and denoted as

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')).$$

Needless to say, the covariance function should be chosen such that the covariance matrix is positive semi-definite. For $x \in \mathbb{R}^D$, a typical category of valid covariance functions, called Gaussian kernels or squared exponentials, has the form

$$k(x, x'; w_0, \ldots, w_D) = w_0 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x'_d)^2}{w_d^2}\right).$$

Such k satisfy stationarity and translational invariance, i.e., $k(x, x') = k(x - x')$. The Euclidean distance describes the correlation between a pair of points. Moreover, the Gaussian kernel is parametrized by hyperparameters $\theta = \{w_0, w_1 \ldots, w_D\}$. Essentially the parameters $w_1, \ldots, w_D$ are interpreted as the length scale on each input dimension [18]. A large w_d indicates almost independence in dimension d. In our problem, the systematic risk factor is unit free in each dimension, i.e., $w_1^2 = \cdots = w_D^2 = \lambda^2$. On the other hand, w_0 measures the overall correlation and is thus called the magnitude scale. We denote the covariance function parametrized by $\theta = \{w_0, \lambda\}$ as

$$k(x, x'; \theta) = w_0 \exp\left(-\frac{1}{2\lambda^2}\sum_{d=1}^{D}(x_d - x'_d)^2\right) \tag{3.4}$$

We review the estimation of θ later. In this section, we take θ to be a given constant.

Now assume the prior for f follows a GP. Let $\mathbf{f}$ be the known function values of the training cases $\mathbf{x}$ and $\mathbf{f}_*$ be the corresponding function values of the testing cases $\mathbf{x}_*$. Because of the joint Gaussian property for any collection of finite points, one can simply formulate the joint training and testing prior as

$$\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\mu}_*\end{bmatrix}, \begin{bmatrix}\Sigma & \Sigma_*\\ \Sigma_*^T & \Sigma_{**}\end{bmatrix}\right) \tag{3.5}$$

where $\boldsymbol{\mu}$ is the training set mean, $\boldsymbol{\mu}_*$ is the testing set mean, Σ is the training set covariance, Σ_* is the training-testing set covariance and Σ_{**} is the testing set covariance.

In some cases, f is a nuisance function, i.e., $\mathbf{f}(\mathbf{x})$ is a hidden state and cannot be observed directly. Instead $\mathbf{y}(\mathbf{x})$ is obtained. The posterior predictive distribution is thus calculated by marginalizing the latent function f as

$$p(f_*|\mathbf{x}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{x}, \mathbf{f}, \mathbf{x}_*)p(\mathbf{f}|\mathbf{x}, \mathbf{y})d\mathbf{f}.$$

Note that the last term is a functional integration, where the domain of the integration is a space of functions and $p(\mathbf{f}|\mathbf{x},\mathbf{y})$ is a probability distribution over the function space. If $\mathbf{y}=\mathbf{f}$ is directly observable, $p(\mathbf{f}|\mathbf{x},\mathbf{y})$ degenerates to the delta function and therefore $p(f_*|\mathbf{x},\mathbf{y},\mathbf{x}_*)=p(f_*|\mathbf{f})$. Needless to say, by looking at the conditional distribution over a multivariate normal, given the training set $\mathbf{f}$, the conditional distribution of $\mathbf{f}_*$ is expressed as

$$\mathbf{f}^*|\mathbf{f}\sim\mathcal{N}(\boldsymbol{\mu}_*+\Sigma_*^T\Sigma^{-1}(\mathbf{f}-\boldsymbol{\mu}),\Sigma_{**}-\Sigma_*^T\Sigma^{-1}\Sigma_*) \tag{3.6}$$

and the posterior process for the test case is still a Gaussian Process:

$$f|\mathcal{D}\sim\mathcal{GP}(m_{\mathcal{D}},k_{\mathcal{D}}) \tag{3.7}$$

with posterior mean and variance

$$m_{\mathcal{D}}(x)=m(x)+\Sigma(x,\mathbf{x})\Sigma^{-1}(\mathbf{f}-\boldsymbol{\mu}) \tag{3.8}$$

$$k_{\mathcal{D}}(x,x')=k(x,x')-\Sigma(x,\mathbf{x})\Sigma^{-1}\Sigma(x',\mathbf{x})^T \tag{3.9}$$

where Σ is the training set covariance given in equation (3.5), $\Sigma(x,\mathbf{x})$ is the covariance between a particular testing point x and the training set $\mathbf{x}$, and similarly for $\Sigma(x',\mathbf{x})$. This posterior Gaussian Process has a smaller variance than the prior, due to the positiveness of the second term in equation (3.9). It is instructive to observe the similarity between the posterior mean and variance in the Gaussian Process Model and the Bayesian linear basis function model in equation (3.2).

Type-II maximum likelihood

So far we have assumed that θ is a given constant. Before a GP is used for prediction, one important step is to fit the GP model by determining the hyperparameter $\theta=\{w_0,\lambda\}$ through training samples. By marginalizing over the latent variable θ, the posterior predictive distribution is

$$p(f_*|\mathbf{f})=\int p(f_*|\mathbf{f},\theta)p(\theta|\mathbf{f})d\theta. \tag{3.10}$$

Needless to say, here we assume f is no longer a nuisance function.

Now the question arises as to whether equation (3.10) is analytically tractable. One approximation method adopts the approach used for point estimation $\hat{\theta}$ and then this simplifies the case where θ is a given constant. A good estimate for θ can be obtained by maximizing the posterior distribution $p(\theta|\mathbf{f})\propto p(\mathbf{f}|\theta)p(\theta)$. A common trick is to maximize the log function, i.e.,

$$\hat{\theta}=\arg\max_{\theta}\{\log p(\mathbf{f}|\theta)+\log p(\theta)\}$$

Since the prior probability for $\mathbf{f}$ given θ is Gaussian with mean $\boldsymbol{\mu}$ and variance Σ,

$$\mathbf{f}|\theta\sim N(\boldsymbol{\mu},\Sigma),$$

the log marginal likelihood is

$$L = \log p(\mathbf{f}|\theta) = -\frac{1}{2}\log|\Sigma| - \frac{1}{2}(\mathbf{f}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{f}-\boldsymbol{\mu}) - \frac{n}{2}\log(2\pi)$$

Note that $\boldsymbol{\mu}$ and Σ may be functions of θ. On the other hand, since we have no particular probability density for the prior of f, $p(\theta)$ can be dropped from the maximization. Hence one approach, suggested by Rasmussen[12], is to choose the hyperparameters to maximize the log marginal likelihood $L = \log p(\mathbf{f}|\theta)$. Thus we choose θ to set the following derivatives to zero:

$$\frac{\partial L}{\partial \theta_m} = (\mathbf{f}-\boldsymbol{\mu})^T\Sigma^{-1}\frac{\partial \boldsymbol{\mu}}{\partial \theta_m} \tag{3.11}$$

$$\frac{\partial L}{\partial \theta_k} = -\frac{1}{2}\operatorname{trace}(\Sigma^{-1}\frac{\partial \Sigma}{\partial \theta_k}) + \frac{1}{2}(\mathbf{f}-\boldsymbol{\mu})^T\Sigma^{-1}\frac{\partial \Sigma}{\partial \theta_k}\Sigma^{-1}(\mathbf{f}-\boldsymbol{\mu}) \tag{3.12}$$

where θ_m is the set of hyperparameters used in the mean function and θ_k is the set of hyperparameters used in the variance function. Given N observations, $\boldsymbol{\mu}$ is an N-dimension vector where each component $\boldsymbol{\mu}_i = m(X_i;\theta_m)$ and Σ is an $N \times N$ matrix where $\Sigma_{i,j} = k(X_i, X_j;\theta_k)$. In our problem, since no hyperparameters are involved in the mean function, equation (3.11) is vacuous. The hyperparameters in the covariance function $\theta_k = \theta = \{w_0, \lambda\}$ are computed by setting equation (3.12) to zero. Rasmussen further proposed a conjugate gradient method as a good approach to solve this optimization problem.

3.3 Bayesian Monte Carlo

Consider again the integral

$$Z = \int f(x)p(x)dx \tag{3.13}$$

where, in our problem, $p(x)$ is the density for the systematic risk factor and $f(x)$ is the loss level given the certain systematic risk factor x. We have seen in Chapter 2, that, by using importance sampling,

$$Z = \int \frac{f(x)p(x)}{q(x)}q(x)dx \approx \frac{1}{N}\sum_n \frac{f(x^{(n)})p(x^{(n)})}{q(x^{(n)})}$$

where the $x^{(n)}$ are random variables with probability density q.

O'Hagan [11], on the other hand, tackles this problem from a Bayesian point of view by treating the integral in equation (3.13) as a Bayesian inference problem. He assumes that Z is a function of f whose prior satisfies the Gaussian Process $\mathcal{GP}(m, k)$. By gradually obtaining observations of $f(x)$, the posterior is still a Gaussian Process, but with new mean $m_{\mathcal{D}}$ and variance $k_{\mathcal{D}}$. The distribution over Z can therefore be implied from this posterior GP.

To make this idea more clear, suppose $\mathcal{D} = \{(x^{(i)}, f(x^{(i)}))|i = 1, \ldots, n\}$ is the set of observations. For a finite number of function values $f(x^{(1)}), f(x^{(2)}), \ldots f(x^{(n)})$, the joint distribution is Gaussian:

$$\mathbf{f} = (f(x^{(1)}), f(x^{(2)}), \ldots f(x^{(n)}))^T \sim \mathcal{N}(m(\cdot), k(\cdot,\cdot)) \tag{3.14}$$

Since f has a GP prior, the posterior $f|\mathcal{D}$ is a GP as well. Since Z is a function of $f(x)$, we have

$$\begin{aligned}\mathbb{E}_{f|\mathcal{D}}[Z] &= \int Z p(f|\mathcal{D}) df \\ &= \int \left(\int f(x) p(x) dx \right) p(f|\mathcal{D}) df \\ &= \int \left(\int f(x) p(f|\mathcal{D}) df \right) p(x) dx = \int \bar{f}_{\mathcal{D}}(x) p(x) dx, \end{aligned} \tag{3.15}$$

where $\bar{f}_{\mathcal{D}}$ is the posterior mean function. From equation (3.8), the posterior mean is given by

$$\bar{f}_{\mathcal{D}}(x) = m(x) + k(x, \mathbf{x}) \Sigma^{-1} (\mathbf{f} - \boldsymbol{\mu}) \tag{3.16}$$

Using a similar approach, the variance is

$$\begin{aligned}\mathbb{V}_{f|\mathcal{D}}[Z] &= \int (Z - \mathbb{E}_{f|\mathcal{D}}[Z])^2 p(f|\mathcal{D}) df \\ &= \int \left[\int f(x) p(x) dx - \int \bar{f}_{\mathcal{D}}(x') p(x') dx' \right]^2 p(f|\mathcal{D}) df \\ &= \int \int \int [f(x) - \bar{f}_{\mathcal{D}}(x)][f(x') - \bar{f}_{\mathcal{D}}(x')] p(f|\mathcal{D}) df p(x) p(x') dx dx' \\ &= \int \int \mathrm{Cov}_{\mathcal{D}}(f(x), f(x')) p(x) p(x') dx dx' \end{aligned} \tag{3.17}$$

where $\mathrm{Cov}_{\mathcal{D}}(f(x), f(x'))$ is the posterior covariance and can be computed from equation (3.9) as

$$\mathrm{Cov}_{\mathcal{D}}(f(x), f(x')) = k(x, x') - k(x, \mathbf{x}) \Sigma^{-1} k(\mathbf{x}, x') \tag{3.18}$$

Rasmussen and Ghahramani [13] further show that if the following conditions are assumed

1. Mean function $m(x) \equiv 0$
2. Density $p(x)$ is Gaussian: $p(x) = \mathcal{N}(b, B)$
3. The kernel K is Gaussian: $K = \mathcal{N}(a_i = x^{(i)}, A = \mathrm{diag}(w_1^2, \ldots, w_D^2))$

then equations (3.15) and (3.17) can be simplified to

$$\mathbb{E}_{f|\mathcal{D}}[Z] = z^T \Sigma^{-1} \mathbf{f} \tag{3.19}$$

$$\mathbb{V}_{f|\mathcal{D}}[Z] = w_0 |2A^{-1}B + I|^{-1/2} - z^T \Sigma^{-1} z \tag{3.20}$$

$$z = w_0 |A^{-1}B + I|^{-1/2} \exp\left(-\frac{1}{2} (a - b)^T (A + B)^{-1} (a - b) \right) \tag{3.21}$$

Specifically, in our problem, $p(x)$ is the standard Gaussian, i.e., $b = 0$ and $B = I$. The kernel matrix A is further assumed to be $\lambda^2 I$, as we stated before. Hence z can be further simplified as

$$\begin{aligned} z &= w_0 |\lambda^{-2} I \cdot I + I|^{-1/2} \exp\left(-\frac{1}{2} a^T (\lambda^2 I + I)^{-1} a \right) \\ &= w_0 (1 + \lambda^{-2})^{-D/2} \exp\left(-\frac{1}{2(1 + \lambda^2)} ||a||_2^2 \right) \end{aligned} \tag{3.22}$$

O'Hagan [11] proposed the Bayesian quadrature for the observation set $\mathcal{D}$ such that the variance can be minimized through a set of fixed optimal points. On the other hand, Rasmussen and Ghahramani [13] choose the optimal importance sampler to minimize the variance. Not surprisingly, to evaluate $\int f(x)p(x)dx$, the optimal sampling distribution is $q^*(x) = cf(x)p(x)$, as we have seen in equation (2.3). We now summarize the Bayesian Monte Carlo method as follows:

1. Sample an observation set $\mathcal{D}$ from the optimal importance sampling distribution using MCMC
2. GP model training and hyperparameter $\theta = \{w_0, \lambda\}$ optimization via equation (3.12)
3. Return the estimator (3.19) using the same points in $\mathcal{D}$

3.4 Drawback of BMC

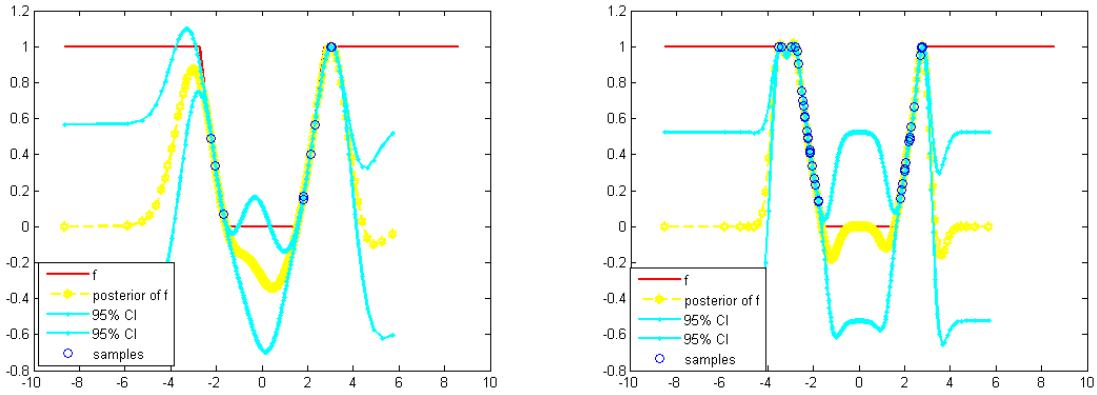


Figure 3.1: A toy example for BMC in 1D. The left panel uses 10 samples; the right panel increases the total number of samples to 40.

BMC, as formulated in the previous section, has a serious drawback. As an illustration of this drawback, consider Figure 3.1. In this toy example, $f(x) = \max(\min(1, 0.2x^2 - 0.5), 10^{-100})$ and $p(x)$ is the pdf of $N(0, 1)$. Thus $f(x)p(x)$ has a two-hump shape. We use the slice sampler to sample the zero variance function $q^*(x) = f(x)p(x)$. The samples are shown as blue circles in Figure 3.1. It is evident that we have better knowledge about the function in regions where the humps occur, whilst the center area is poorly sampled. As a result, f is approximated badly in the center area. Notice that increasing the number of samples doesn't significantly improve the approximation in the center area. Therefore, a major drawback is that Bayesian Monte Carlo cannot guarantee positiveness and negative values make no sense at all in this context.

This drawback of BMC can lead to very poor results. If we are dealing with an extreme loss level, samples from MCMC will locate in the tail. We would therefore have a large uncertainty around the origin where the probability density is large. If, as a result, the posterior of the function is negative in the uncertain region, the approximation to the loss probability may be negative.

Several ways are proposed to overcome this issue. To start with, one could model the Gaussian Process on a nonnegative space rather than the function itself. Such an approach, suggested by Osborne et al. [8], is called Active Learning Using Bayesian Quadrature. We implemented this method, but, unfortunately, for our problem, the results are not as good as what we expected. One possible reason for

this is that, in this sophisticated method, one needs to train three Gaussian processes, which increases the running time significantly. On the other hand, and more importantly, at the very extreme cases, such as a 99.99% VaR calculation, negative values may still occur. Thus non-negativeness cannot be guaranteed.

Another possible approach is more practical. We extend the BMC method by choosing $\mathcal{D}$ to be a mixture of samples from the zero variance function and a standard Gaussian, to cover the hole in the middle. We would use more samples from the standard Gaussian when the dimension of the problem is higher. This approach is illustrated in Figure 3.2.

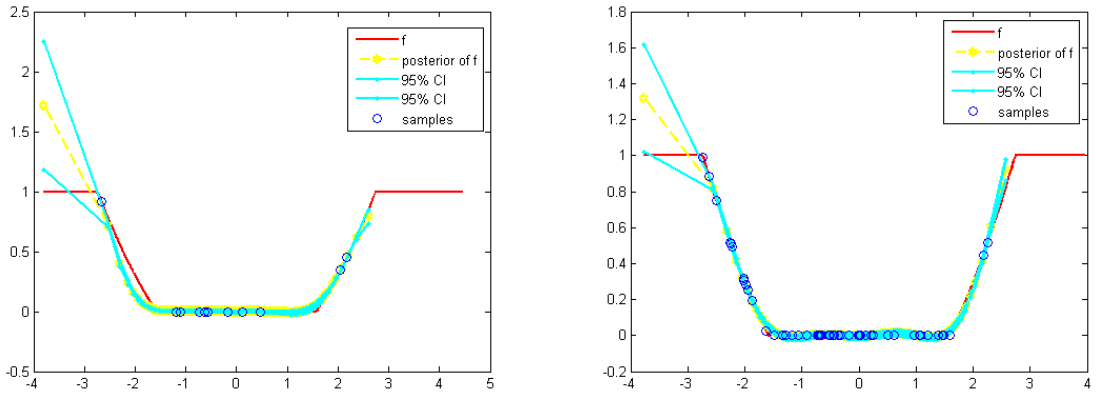


Figure 3.2: A toy example for BMC in 1D with better estimation. The left panel uses 10 samples, where 2 samples are from MCMC and 8 samples are from the standard normal $N(0,1)$; the right panel increases the total number of samples to 50, where 10 samples are from MCMC and 40 are from $N(0,1)$.

3.5 Hybrid BMC

We now proposed a hybrid Bayesian Monte Carlo that overcomes the drawback discussed in the previous section. According to the three-sigma rule, i.e., in a normal distribution, $P(\mu - 3\sigma \leq x \leq \mu + 3\sigma) \approx 0.9973$, where μ is the mean and σ is the standard deviation, we have to guarantee that the posterior of f is a good approximation of f within the 3σ interval. Let's consider a new toy example. The left panel in Figure 3.3 shows the zero variance function at a high loss level as we have seen in Figure 2.1c while the right panel shows $f(x)p(x)$, where $f(x) = \max(\min(1, 0.2(x-0.5)^2 - 1), 10^{-100})$ and $p(x)$ is the pdf of $N(0,1)$.

The hybrid BMC combines the key ideas from Chapter 2 and this Chapter by using the approximation to the zero variance function developed in Chapter 2 with the Bayesian Monte Carlo approach. Since the squared exponential covariance function is infinitely differentiable, the Gaussian Process must be very smooth [12]. However, in our problem, f itself is not very smooth. Since modeling f directly as a Gaussian Process gives a poor result, we apply the importance sampling technique

$$\int f(x)p(x)dx = \int \frac{f(x)p(x)}{q(x)} q(x)dx = \int h(x)q(x)dx \tag{3.23}$$

where $p(x)$ is the pdf of $N(0,1)$ and $q(x)$ is the pdf of $N(\mu, \sigma)$. Then, h, rather than f, is modeled as a Gaussian Process. Note that $h(x)$ has the following two important properties. Firstly $h(x)$ is smooth

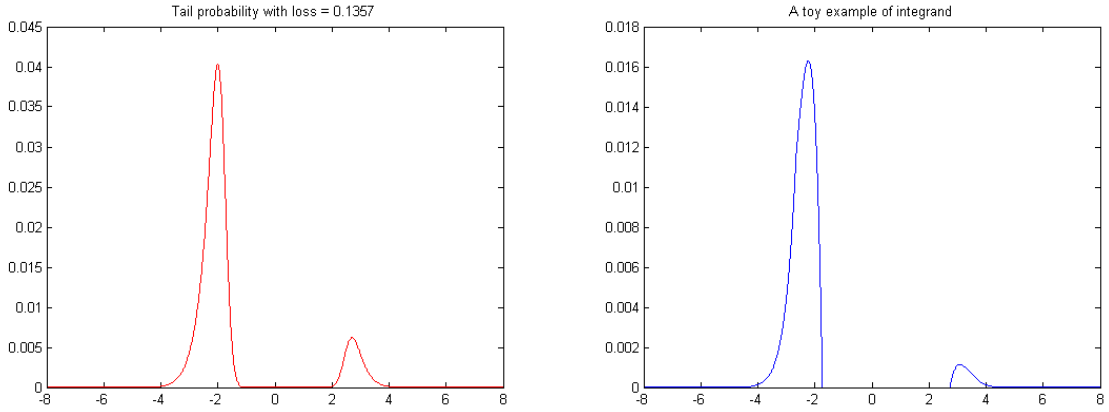


Figure 3.3: A toy example for a high loss level in 1D. The left panel shows a sampled zero variance function, while the right panel shows $f(x)p(x)$, where $f = \max(\min(1, 0.2(x-0.5)^2 - 1), 10^{-100})$ and $p(x)$ is the pdf of $N(0,1)$.

enough that a Gaussian Process with a squared exponential covariance function is a good model. More importantly, within the 3σ interval $\mu \pm 3\sigma$, $h(x)$ is bounded away from zero, assuming μ and σ are chosen appropriately. Hence, it is much easier to preserve non-negativity with h than f.

In this toy example, $\mu = -2.3711$ and $\sigma = 0.1234$ are obtained through maximizing the log likelihood of the data sampled from the slice sampler. In Figure 3.4, $h(x)$ is plotted as the solid blue line and the dashed green line is the pdf of $N(\mu, \sigma)$ over the range $\mu \pm 3\sigma$. The posterior of h comes from sampling from $N(\mu, \sigma)$ and is represented by the yellow line. With just 10 samples, the left panel shows that this transformation gives a better posterior than modeling f directly in Figure 3.1. By adding more samples, the posterior gives a much more accurate estimation.

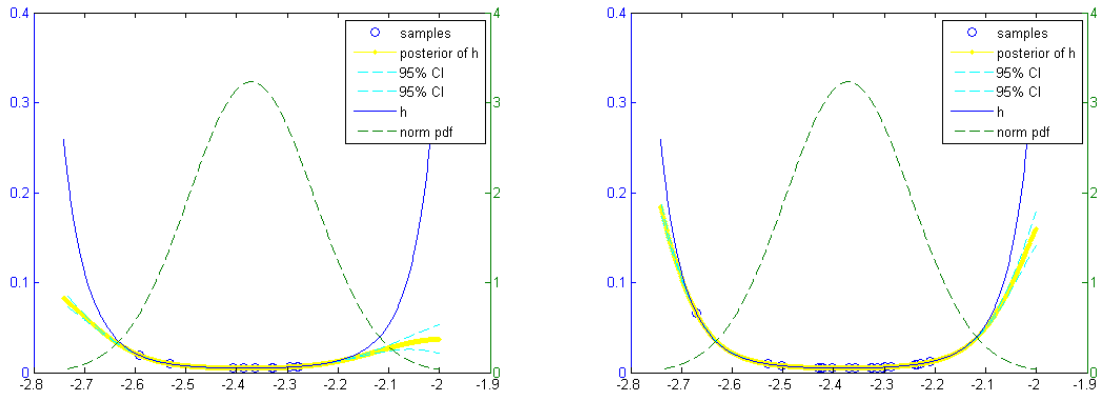


Figure 3.4: A toy example for the hybrid BMC in 1D. The importance sampler is $N(-2.3711, 0.1234)$. The left figure shows the posterior of the transformed integrand with 10 observations, while the right one uses 20 observations.

Note that this approach becomes effective when the loss level is high enough that a normal distribution $N(\mu, \sigma)$ is successfully learned and can approximate the zero variance function well. In a bad case, such that the importance sampler parameters are not successfully learned (e.g., $\mu = -1.7678$ and $\sigma = 3.5648$),

the importance sampling technique doesn't help transform the difficult-to-model integrand into the one we see in Figure 3.4. We show two bad modeling cases below in Figure 3.5.

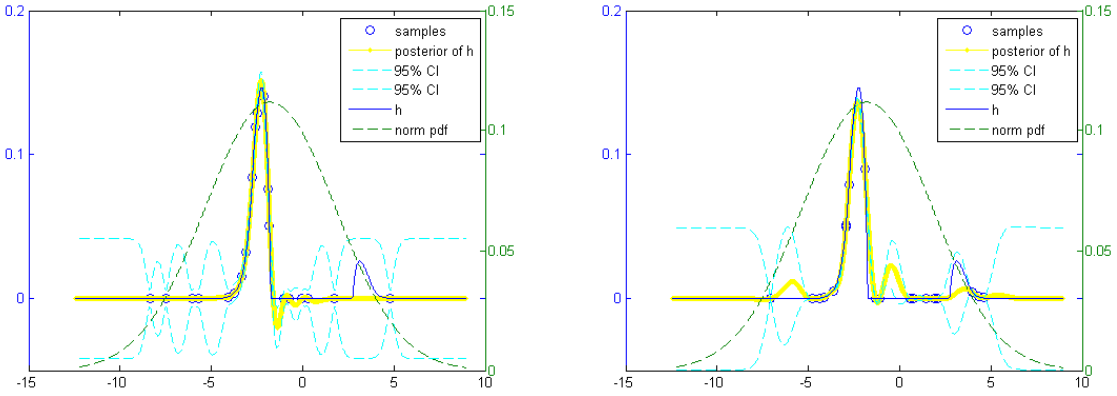


Figure 3.5: A toy example showing the effects of a poor importance sampler. The importance sampler is $N(1.7678, 3.5648)$.

In such a circumstance, we resort to the Gaussian Mixture Model again. Assume the total number of components K is predetermined and $q(x) = \sum_{k=1}^{K} \alpha_k \phi(x; \mu_k, \Sigma_k)$ subject to $\alpha_k \geq 0$ and $\sum_{k=1}^{K} \alpha_k = 1$. The rightmost term in equation (3.23) becomes

$$\int h(x)q(x)dx = \int h(x) \sum_{k=1}^{K} \alpha_k \phi(x; \mu_k, \Sigma_k)dx = \sum_{k=1}^{K} \alpha_k \int h(x)\phi(x; \mu_k, \Sigma_k)dx \tag{3.24}$$

where the last term is nothing but the weighted summation of posterior integrals. For each component, $h(x)$ is expected to be smooth and locally modeled well through a Gaussian Process. In Figure 3.6, we use 15 training samples from a slice sampler for each component and obtain the mixture model shown by the green dashed line. The posterior of h and its confidence intervals are also shown.

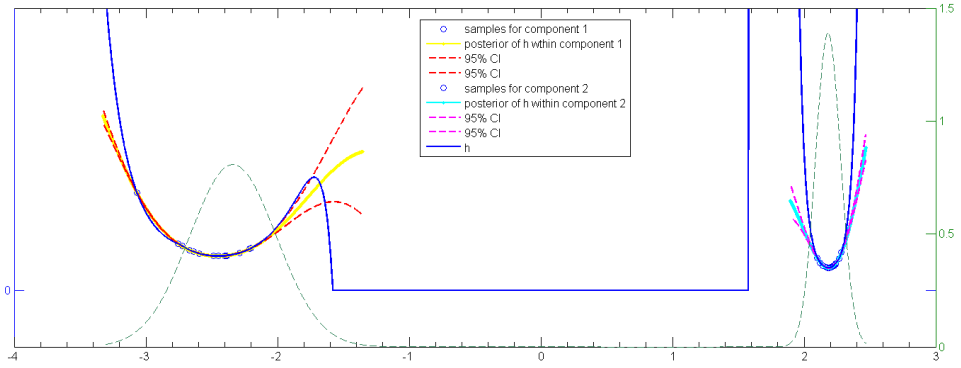


Figure 3.6: A toy example illustrating a 2 mode importance sampler

Chapter 4

Two Step Importance Sampling

Importance sampling (IS) is a common technique used in sampling theory. Rather than sampling directly from the original distribution, one could sample from another distribution which is absolutely continuous with respect to the original one. Then one must correct the MC estimation by multiplying each MC sample by the associated likelihood function. For this particular problem with the appearance of two risk factors $\mathcal{Z}$ and $\mathcal{E}$, this sampling technique becomes fairly tricky.

Glasserman and Li [3] suggest a two-step IS procedure for the digital credit state model. In the previous section, the conditional default probability for obligor n is denoted as $p_n^D(\mathbf{z})$ and default indicator is $\mathbb{1}_n^D(\mathbf{z})$. In the Gaussian Coupula model, obligors are independent of each other conditional on $\mathbf{z}$. The joint probability mass function of the default indicator is therefore

$$f_{\mathbb{1}_1^D(\mathbf{z}),\ldots,\mathbb{1}_N^D(\mathbf{z})}(x_1,\ldots,x_N) = \prod_{n=1}^{N} p_n^D(\mathbf{z})^{x_n}(1-p_n^D(\mathbf{z}))^{1-x_n} \tag{4.1}$$

where each x_n is either 0 or 1. The conditional total loss is

$$L_N(\mathbf{z},\mathcal{E}) = \sum_{n=1}^{N} h_n \mathbb{1}_n^D(\mathbf{z})$$

For importance sampling, we find $q_n(\mathbf{z})$ absolutely continuous with respect to the original default probabilities $p_n^D(\mathbf{z})$ for $n = 1, 2, \ldots, N$. Then

$$\begin{aligned}
\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\} &= \mathbb{E}[\mathbb{P}(L_N(\mathcal{Z},\mathcal{E}) > l | \mathcal{Z} = \mathbf{z})] \\
&= \mathbb{E}[\mathbb{E}[\mathbb{1}_{L_N(\mathcal{Z},\mathcal{E})>l} | \mathcal{Z} = \mathbf{z}]] \qquad (4.2) \\
&= \mathbb{E}\left[\tilde{\mathbb{E}}\left[\mathbb{1}_{L_N(\mathcal{Z},\mathcal{E})>l} \prod_{n=1}^{N} \left(\frac{p_n^D(\mathcal{Z})}{q_n(\mathcal{Z})}\right)^{\mathbb{1}_n^D(\mathcal{Z})} \left(\frac{1-p_n^D(\mathcal{Z})}{1-q_n(\mathcal{Z})}\right)^{1-\mathbb{1}_n^D(\mathcal{Z})} \middle| \mathcal{Z} = \mathbf{z}\right]\right] \qquad (4.3)
\end{aligned}$$

where $\mathbb{E}[\cdot|\mathcal{Z} = \mathbf{z}]$ in equation (4.2) is the expectation with respect to the original default probabilities $p_1^D(\mathbf{z}), \ldots, p_N^D(\mathbf{z})$, $\tilde{\mathbb{E}}[\cdot|\mathcal{Z} = \mathbf{z}]$ in equaton (4.3) is the expectation using the new default probabilities $q_1(\mathbf{z}), \ldots, q_N(\mathbf{z})$ and the default indicators $\mathbb{1}_n^D(\mathbf{z})$'s are sampled from this new probability density.

4.1 Conditional Importance Sampling

Glasserman and Li [3] suggested using the following exponential twisting to compute the $q_n(\mathbf{z})$:

$$q_n(\mathbf{z}) = p_{n,\theta(\mathbf{z})} = \frac{p_n^D(\mathbf{z})e^{\theta(\mathbf{z})h_n}}{1 + p_n^D(\mathbf{z})(e^{\theta(\mathbf{z})h_n} - 1)} \tag{4.4}$$

where $\theta(\mathbf{z})$ is a parameter to be chosen to reduce the variance. Note that $p_n^D(\mathbf{z})$ and $p_{n,\theta(\mathbf{z})}$ are equivalent in the sense that $p_n^D(\mathbf{z}) = 0$ if and only if $p_{n,\theta(\mathbf{z})} = 0$. Moreover, if $\theta(\mathbf{z}) = 0$, then $p_{n,\theta(\mathbf{z})} = p_n^D(\mathbf{z})$, i.e. no twist is performed. In addition, the likelihood ratio is

$$\prod_{n=1}^{N} \left(\frac{p_n^D(\mathbf{z})}{p_{n,\theta(\mathbf{z})}}\right)^{\mathbb{1}_n^D(\mathbf{z})} \left(\frac{1 - p_n^D(\mathbf{z})}{1 - p_{n,\theta(\mathbf{z})}}\right)^{1-\mathbb{1}_n^D(\mathbf{z})} = \exp(-\theta(\mathbf{z})L_N(\mathbf{z},\mathcal{E}) + \psi(\theta(\mathbf{z}))) \tag{4.5}$$

where

$$\psi(\theta(\mathbf{z})) = \sum_{n=1}^{N} \log\left(1 + p_n^D(\mathbf{z})(e^{\theta(\mathbf{z})h_n} - 1)\right).$$

Therefore the inner expectation satisfies

$$\mathbb{E}[\mathbb{1}_{L_N(\mathbf{z},\mathcal{E})>l}] = \tilde{\mathbb{E}}[e^{-\theta(\mathbf{z})L_N(\mathbf{z},\mathcal{E})+\psi(\theta(\mathbf{z}))}\mathbb{1}_{L_N(\mathbf{z},\mathcal{E})>l}]$$

and the second moment associated with the expectation on the right is

$$M_2(\theta(\mathbf{z})) = \tilde{\mathbb{E}}[e^{-2\theta(\mathbf{z})L_N(\mathbf{z},\mathcal{E})+2\psi(\theta(\mathbf{z}))}\mathbb{1}_{L_N(\mathbf{z},\mathcal{E})>l}] \leq \exp(-2\theta(\mathbf{z})l + 2\psi(\theta(\mathbf{z}))) \tag{4.6}$$

Note that the probability density twisting happens only when $L_N(\mathbf{z},\mathcal{E}) > l$. To find the better importance sampling probability, the consequential optimization is derived by minimizing the upper bound. The inner level computation algorithm is given by:

1. Calculate the conditional default probabilities $p_n^D(\mathbf{z}), n = 1, \ldots, N$.

2. Set $\theta(\mathbf{z}) = 0$ if

$$L_N(\mathbf{z},\mathcal{E}) \equiv \sum_{n=1}^{N} p_n^D(\mathbf{z})h_n \leq l$$

otherwise, set $\theta(\mathbf{z})$ equal to the unique solution of the following problem:

$$\theta(\mathbf{z}) = \arg\min_{\theta}\{-2\theta l + 2\psi(\theta)\} \qquad \text{where} \qquad \psi(\theta) = \sum_{n=1}^{N} \log\left(1 + p_n^D(\mathbf{z})(e^{\theta h_n} - 1)\right) \tag{4.7}$$

3. Generate multiple samples of default indicators $\mathbb{1}_1^D(\mathbf{z}), \ldots, \mathbb{1}_N^D(\mathbf{z})$ from the twisted conditional default probabilities

$$p_{n,\theta(\mathbf{z})} = \frac{p_n^D(\mathbf{z})e^{\theta(\mathbf{z})h_n}}{1 + p_n^D(\mathbf{z})(e^{\theta(\mathbf{z})h_n} - 1)}, \qquad n = 1, \ldots, N.$$

and compute the loss $L_N(\mathbf{z},\mathcal{E}) = h_1\mathbb{1}_1^D(\mathbf{z}) + \cdots + h_N\mathbb{1}_N^D(\mathbf{z})$ for each sample

4. Return the inner level estimator of $\mathbb{P}(L_N(\mathcal{Z},\mathcal{E}) > l|\mathcal{Z}=\mathbf{z})$ by averaging

$$\exp\{-\theta(\mathbf{z})L_N(\mathbf{z},\mathcal{E}) + \psi(\theta(\mathbf{z}))\}\mathbb{1}_{L_N(\mathbf{z},\mathcal{E})>l}.$$

4.2 Outer level Importance Sampling

The importance sampling technique is also applied to the outer level. Rather than sampling the systemic risk factor Z from $N(0,I)$, one can sample from a mean shifted normal $N(\mu,I)$. Generally speaking, the mode μ should be selected such that the probability density achieves its largest value, i.e. μ should be the solution to the optimization problem:

$$\mu = \arg\max_{\mathbf{z}} \mathbb{P}(L_N(\mathcal{Z},\mathcal{E}) > l|\mathcal{Z}=\mathbf{z})e^{-\mathbf{z}^T\mathbf{z}/2}.$$

Several approaches are discussed in Glasserman and Li[3], among which the tail bound approximation is used in their paper. Therefore we also adopt this approximation in our numerical experiments. This method resorts to an approach similar to that used in equation (4.6), namely

$$\mathbb{P}(L_N(\mathcal{Z},\mathcal{E}) > l|\mathcal{Z}=\mathbf{z}) = \tilde{\mathbb{E}}[e^{-\theta(\mathbf{z})L_N(\mathbf{z},\mathcal{E})+\psi(\theta(\mathbf{z}))}\mathbb{1}_{L_N(\mathbf{z},\mathcal{E})>l}] \leq e^{-\theta((\mathbf{z})l+\psi(\theta((\mathbf{z}))} \tag{4.8}$$

To minimize the upper bound on the right side of (4.6), we solve the optimization problem

$$\begin{aligned}\mu &= \arg\max_{\mathbf{z}}\{-\theta(\mathbf{z})l + \psi(\theta(\mathbf{z})) - \frac{1}{2}\mathbf{z}^T\mathbf{z}\} \\ &= \arg\max_{\mathbf{z}}\{\min_{\theta(\mathbf{z})\in\Theta}\{-\theta(\mathbf{z})l + \psi(\theta(\mathbf{z}))\} - \frac{1}{2}\mathbf{z}^T\mathbf{z}\}\end{aligned} \tag{4.9}$$

Notice that this maximization actually involves a nested optimization, as θ is determined through a conditional minimization in equation (4.7). We list the overall steps in Algorithm 5.

Algorithm 5 Two level Importance Sampling

1: Select a new mean μ for $\mathcal{Z}$ by solving $\arg\max_{\mathbf{z}} \mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l|\mathcal{Z}=\mathbf{z}\}e^{-\mathbf{z}^T\mathbf{z}/2}$ in equation (4.9)
2: Sample systemic risk $\mathbf{z}$'s from $N(\mu,\mathbf{I})$
3: **for** each systemic risk $\mathbf{z}$ **do**
4: Compute conditional default probability $p_n^D(\mathbf{z})$
5: Compute the twist factor $\theta(\mathbf{z})$ in equation (4.7) if possible
6: Compute the twisted conditional default probability $p_{n,\theta(\mathbf{z})}$
7: Compute the inner level conditional estimator $\mathbb{P}(L_N(\mathcal{Z},\mathcal{E}) > l|\mathcal{Z}=\mathbf{z})$ by sampling conditional default indicators $\mathbb{1}_1^D(\mathbf{z}),\ldots,\mathbb{1}_N^D(\mathbf{z})$
8: **end for**
9: **return** Estimator $\mathbb{P}(L_N(\mathcal{Z},\mathcal{E}) > l)$ by averaging each inner level conditional estimator

This method has several drawbacks. First, the optimization problem (4.9) is hard to solve. Second, the tail bound approximation proposed doesn't reduce the complexity a lot due to a nested optimization structure. Third, for each systemic factor $\mathbf{z}$ sampled, there is an associated optimization problem of finding the twisted parameter θ. As a result, when the algorithm is implemented, it is not as fast as we initially expected. Furthermore, the risk model we used here is a simple one with digital credit states only, i.e. an obligor is either in default or not. Due to the fact that equation (4.3) takes advantage

of independent binomial distributions, it is not clear how this approach could be extended to multiple credit states. The multiple credit states problem could be so complicated that the exponential twisting technique is no longer applicable.

Chapter 5

Numerical Experiments

In this chapter, we compare all the methods we have discussed so far: the two-step Importance Sampling (IS) method proposed by Glasserman and Li [3] and reviewed in Chapter 4 (GLM), the zero variance function estimation approach developed in Chapter 2 which we refer to as SGk, where k is the number of terms in the Gaussian Mixture Model probability density function (2.17), the hybrid Bayesian Monte Carlo method developed in Chapter 3 (BMC), as well as the two crude Monte Carlo approaches outlined in Section 1.2, a two level simple crude Monte Carlo (2LvlMC) and a one level crude Monte Carlo (CrudeCLT) to evaluate equation (1.13) based on CLT. For SGk, we focus on the cases $k = 1$ and $k = 2$. To generate samples from the zero variance function used in SG1, SG2 and BMC, we choose the slice sampler implemented in the MATLAB built-in function *slicesample*. Two input arguments of *slicesample* are worth discussing. Thin level h is a positive integer; the slice sampler discards every $h - 1$ samples and returns the next. Burn-in b is a nonnegative integer, indicating the number of initial samples to discard; the samples that are actually used start at sample $(b + 1)$. Since GLM uses an exponential twisting based on the digital-credit-state model, we conduct our experiment under the credit migration matrix shown in Section 1.2.2 in order to be compatible with the setting used in Glasserman and Li [3]. The MATLAB built-in function *fminunc* is used to solve equation (4.9).

5.1 Configurations

In some applications, we need to compute $\mathbb{P}\{L_N(\mathcal{Z}, \mathcal{E}) > l\}$ just once, however, in others, we need to compute $\mathbb{P}\{L_N(\mathcal{Z}, \mathcal{E}) > l\}$ for several l's that are close together. For example, to compute value-at-risk (VaR), we could use bisection to find the value $l = VaR$ such that $\mathbb{P}\{L_N(\mathcal{Z}, \mathcal{E}) > VaR\} = 1 - \delta$, where δ is the desired level associated with VaR (e.g., $\delta = 99.9\%$). Note that the overall speed of such a VaR calculation does not only depend on the number of samples used, but also on the overhead calculations (i.e., associated preliminary calculations) such as optimization, slice sampling and data clustering. However, it's not necessary to repeat the overhead in each bisection iteration. For example, for the method discussed in Chapter 2, the zero variance function estimation for SGk could be computed once for all the bisection iterations. This zero variance function estimation step could be counted as overhead in the numerical results. Needless to say, after several iterations, one could redo the zero variance function estimation procedure and use an updated importance sampler for better accuracy. In GLM, the overhead refers to solving the nested optimization problem to find the shifted mean. Therefore,

in the numerical results that we report in this chapter, we list an overhead for a method as well as the cost to compute $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\}$ once, excluding the overhead.

Two sets of experiments are conducted in this chapter. The first set computes $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\}$, where l is set to be the 99.9% VaR. The second one mimics applying bisection to solve for VaR, by working out the tail probabilities for a set of l's corresponding to 90%, 95%, 99%, 99.5%, 99.9%, 99.95%, 99.99% VaRs. This test is performed for $S = 5, 10, 20, 30$. For both sets of numerical experiments, in the first stage, we use a two-level crude Monte Carlo (2LvlMC) method to compute an accurate approximation to the solution of these test cases to use in the computation of the error associated with the methods tested in this chapter. In this two-level MC method, we use 5000 S-dimension systemic factor samples and 2500 N-dimension individual idiosyncratic factor samples, or equivalently $5000(S + 2500N)$ individual 1 dimensional normal samples. Furthermore, we sample 400,000 S-dimension systemic factors to accurately evaluate equation (1.13). There is a difference between these two 'true' values that represents a bias incurred by our new methods, SGk and BMC, as well as CrudeCLT, as they are designed to approximate the solution of (1.13). We use the benchmark 2LvlMC method to compute VaRs associated with our test portfolio described below. In the second stage, we examine all the techniques listed in the opening paragraph of this chapter, by setting the loss level l to the values of VaR computed in the first stage and comparing the loss probability $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\}$ for each VaR.

Binary Credit States

As noted above, to be consistent with Glasserman and Li [3], in our test results, we use a synthetic portfolio based on digital credit states (i.e., the credit state is either default or no default). The marginal default probability and the loss at default of the kth obligor are

$$p_k = 0.01 \cdot (1 + \sin(16\pi k/N)), \qquad k = 1, \ldots, N$$

$$LGC_k^D = (\lceil 5k/N \rceil)^2, \qquad k = 1, \ldots, N.$$

The correlation of the jth risk factor component for the kth obligor, β_{kj}, is generated independently and uniformly from $(-1/\sqrt{S}, 1/\sqrt{S})$, where S is the dimension of the systematic risk factor. Moreover, the exposure-at-default and the corresponding exposure weight of obligor k are

$$EAD_k \sim \text{Unif}(0.5, 1.5), \qquad \omega_k = \frac{EAD_k}{\sum_{n=1}^{N} EAD_n}.$$

5.2 99.9% VaR Computation

For the first comparison of all the methods considered in this paper, we use each method listed in the opening paragraph of this chapter to compute the 99.9% VaR with dimension $S = 20$ and $N = 2500$ obligors. This example is quite relevant for many financial institutions.

We use SG$k(m, n)$ to indicate the dataset size in the numerical experiments for SGk, where m is the number of training MCMC samples (thin level is 3 and the first 10% of the samples are burned in) and n is the number of testing samples from the GMM importance sampler; BMC(m, n), where m is the number of samples from the slice sampler for the zero variance function estimation and n is the number of training samples for the Gaussian Process; GLM(m, n), where m is the number of samples in the outer level and n is the number of samples in the inner level; 2LvlMC(m, n), where m and n are the number of

samples for the outer and inner levels, respectively; and CrudeCLT(m), where m MC samples are applied to equation (1.13). The benchmark 2LvlMC(5000, 2500) is used to compute an accurate approximation to $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\}$ and CrudeCLT(400,000) is used to compute an accurate approximation to equation (1.13). Of course, we use much smaller m and n in the numerical experiments comparing 2LvlMC(m, n) and CrudeCLT(m) to SGk, BMC and GLM.

We first compare the following six methods: SG1(300, 600), SG2(300, 600), GLM(90, 100), BMC(150, 200), 2LvlMC(300, 200) and CrudeCLT(20000). To reduce the randomness in our numerical results, we repeat the test for each method 10 times and average the results from 10 iterations. We also check if the variance is reduced effectively by looking at the variance of the 10 iterations. We run a similar second round of tests with SG1(600, 2000), SG2(600, 2000), GLM(300, 100), BMC(150, 400), 2LvlMC(400, 250) and CrudeCLT(40000). Lastly we run a similar third round of tests with SG1(800, 4000), SG2(800, 4000), GLM(550, 100), BMC(150, 600), 2LvlMC(500, 300) and CrudeCLT(60000). It's worth mentioning that we picked the number of samples for the six methods tested carefully, as will become clear from the discussion of the test results below.

The average running time for each iteration of 2LvlMC(300, 200) and CrudeCLT(20000) is about 5 seconds. For 2LvlMC(400, 250) and CrudeCLT(40000), it's about 8 to 10 seconds and, for 2LvlMC(500, 300) and CrudeCLT(60000), it is about 13 to 16 seconds. The exact statistics for these two crude Monte Carlo methods are listed in Table 5.1. The total running time in the fourth column in Table 5.1 is the total computational time for the overhead and all 10 iterations. Comparing the crude two-level MC method and crude CLT method for each round, we see that the mean of CrudeCLT is much closer to the true value[1] (i.e., 0.1%), than is the mean 2LvlMC in two of the three tests. This also can be seen from the relative error percentage. Moreover, the variance from 10 iterations is reduced by utilizing CLT. For roughly the same amount of time, the variance of 2LvlMC(500, 300) for 10 iterations is about 4 times larger than that of CrudeCLT(60000). However, transforming a two-level problem into a one-level one by adopting CLT has some disadvantages. Consider the 95% confidence interval for CrudeCLT(60000). The lower point is $\mu - 1.96\frac{\sigma}{\sqrt{n}} \approx$ 8.38E-04 and the upper point is $\mu + 1.96\frac{\sigma}{\sqrt{n}} \approx$ 9.13E-04. That is, there is a bias in the results of CrudeCLT that arise from the approximation associated with the use of Central Limit Theorem, as noted earlier in this chapter. The value from CrudeCLT with 400,000 samples is about 8.894E-04, which gives a relative error of about 10% for all the new methods based on CLT. Note that Han [5] derived a confidence interval and error analysis for the CLT approximation. The 10% relative error here is larger than we would have expected from his results. This may be because we are working with a binary credit states model. As a result, the credit state transition is not smooth and so the systemic risk factor surface is not smooth either.

Next, we look at the numerical results for GLM. For all the three runs, GLM(90, 100), GLM(300, 100) and GLM(550, 100), GLM takes a long time and its performance is poor. When the problem dimension is large ($S = 20$) and we focus on the tail probability (99.9% VaR), the optimization problem in equation (4.9) is hard to solve. As a result, *fminunc* may fail to return an optimum (or even a local optimum) depending on the initial problem configuration and the loss level. When *fminunc* fails, the importance sampler associated with GLM is not very effective. With this poor importance sampler, we do not get accurate results with the number of samples we have chosen to use. In other words, without identifying black swan events successfully, the integral is undervalued, or, financially speaking, the VaR

[1] True value is the probability $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\} = 0.001$ where l is the loss level associated with the VaR at the 99.9% level and VaR was calculated by the accurate benchmark two-level crude MC method.

is undervalued, as we can see from Table 5.1, where the mean for GLM is much smaller than the true value. What's more, in each iteration, the same optimization problem in equation (4.9) is repeatedly solved.

	avg. overhead (in sec)	avg. iter. time (in sec)	total time (in sec)	mean	variance	Rel Err
2LvlMC(300, 200)	-	5.0578	50.5781	6.600E-04	5.453E-04	34.00%
CrudeCLT(20000)	-	5.3213	53.2127	8.694E-04	9.989E-05	13.06%
GLM(90, 100)	-	18.7848	187.8483	6.949E-05	9.023E-06	93.05%
GLM*(90, 100)	10.9045	2.2486	33.3905	2.524E-04	1.630E-04	74.76%
SG1(300, 600)	4.0786	0.3667	7.7456	9.814E-04	3.281E-04	1.86%
SG2(300, 600)	4.3483	0.4564	8.9123	5.069E-04	2.111E-04	49.31%
BMC(150, 200)	2.5590	0.5750	8.3091	1.663E-03	2.271E-03	66.28%
2LvlMC(400, 250)	-	8.4933	84.9328	1.057E-04	9.871E-04	5.70%
CrudeCLT(40000)	-	10.6003	106.0028	8.821E-04	6.444E-05	11.79%
GLM(300, 100)	-	22.1172	221.1718	5.927E-05	3.074E-04	94.07%
GLM*(300, 100)	10.9045	4.5013	55.9175	1.838E-04	9.597E-05	81.62%
SG1(600, 2000)	6.9483	1.2219	19.1673	8.475E-04	4.402E-05	15.25%
SG2(600, 2000)	7.4900	1.3435	20.9253	8.762E-04	2.054E-04	12.38%
BMC(150, 400)	2.5590	1.8711	21.2704	1.061E-03	9.185E-04	6.13%
2LvlMC(500, 300)	-	12.7530	127.5302	6.960E-04	2.787E-04	30.40%
CrudeCLT(60000)	-	15.8709	158.7093	8.756E-04	6.092E-05	12.44%
GLM(550, 100)	-	26.2088	262.0876	8.169E-05	4.004E-05	91.83%
GLM*(550, 100)	10.9045	6.7281	78.1855	3.256E-04	3.056E-04	67.44%
SG1(800, 4000)	10.0590	2.4045	34.1042	8.770E-04	3.486E-05	12.30%
SG2(800, 4000)	10.2165	2.7216	37.4325	8.490E-04	9.244E-05	15.10%
BMC(150, 600)	2.5590	3.8148	40.7068	7.686E-04	5.138E-04	23.14%

Table 5.1: 99.9% VaR comparison: computing time, mean of $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\}$, variance and relative error for the methods tested for $S = 20$ and $N = 2500$. The computation is broken into two stages: overhead and importance sampling. The true value of $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\}$ is about 1.00E-03. The true value of the integral in equation (1.13) is about 8.894E-04.

One might say that it is not a fair comparison for GLM if the shifted mean for GLM is not successfully found. Therefore, we ran a second comparison for GLM, in which GLM first uses a lower loss level (say 90% VaR) for which the two-level optimization problem in equation (4.9) associated with GLM is always solved. Then we reuse the GLM parameters that we find for the lower loss level for the runs with the higher loss level. That is, in each iteration, we avoid the overhead of recomputing the GLM parameters, but focus on the exponential twisting and re-sampling. We denote this method as GLM*(m, n).

Similarly for SG1, SG2 and BMC, the zero variance function estimation step (i.e., the overhead) is done just once and then the parameters are reused without re-computation in 10 MC iterations, each of which computes $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\}$. However, SG1, SG2 and BMC have robust zero variance function estimations for all the loss levels we tested. In our tests, the GLM* overhead is calculated at a lower loss level while overheads in our new methods are calculated at the higher loss level, i.e., l is associated with the VaR at the 99.9% level. As stated earlier in this chapter, the valuation iteration is repeated several times in the bisection method to compute VaR. Hence, we focus more on the total running time listed in column four in Tables 5.1 and 5.2, which is the overhead cost for one run (in column two) plus 10 times the cost of one MC iteration (listed in column three).

Results for the four methods, GLM*, SG1, SG2 and BMC, are summarized in Table 5.1. Note that

	avg. overhead (in sec)	avg. iter. time (in sec)	total time time (in sec)	mean	variance	Rel Err
2LvlMC(300, 200)	-	6.0668	60.6682	9.650E-04	7.583E-05	3.50%
CrudeCLT(20000)	-	5.7883	57.8827	8.400E-04	1.054E-04	16.00%
GLM(90, 100)	-	16.1508	161.5075	6.415E-05	6.664E-05	93.58%
GLM*(90, 100)	11.3869	2.2491	33.8779	3.814E-04	4.279E-04	61.86%
SG1(300, 600)	4.3332	0.3684	8.0172	8.552E-04	4.756E-04	14.48%
SG2(300, 600)	4.4749	0.3962	8.4369	3.247E-04	2.621E-04	67.53%
BMC(150, 200)	2.7381	0.6301	9.0391	9.040E-04	8.498E-04	9.60%
2LvlMC(400, 250)	-	10.1933	101.9330	7.290E-04	7.211E-05	27.10%
CrudeCLT(40000)	-	11.5781	115.7812	8.682E-04	5.114E-05	13.18%
GLM(300, 100)	-	19.6860	196.8603	1.888E-03	5.594E-03	88.85%
GLM*(300, 100)	11.3869	4.4247	55.6339	1.753E-04	8.608E-05	82.47%
SG1(600, 2000)	8.2106	1.2607	20.8176	8.560E-04	8.637E-05	14.40%
SG2(600, 2000)	8.6756	1.3586	22.2616	8.640E-04	1.804E-04	13.60%
BMC(150, 400)	2.7381	1.7113	19.8511	6.445E-04	6.176E-04	35.55%
2LvlMC(500, 300)	-	15.1958	151.9579	8.527E-04	5.716E-04	14.73%
CrudeCLT(60000)	-	17.4669	174.6687	8.547E-04	4.334E-05	14.53%
GLM(550, 100)	-	23.8920	238.9204	3.275E-04	7.244E-04	67.25%
GLM*(550, 100)	11.3869	6.5112	76.4989	3.150E-04	2.459E-04	68.50%
SG1(800, 4000)	11.7201	2.4094	35.8141	8.573E-04	4.434E-05	14.27%
SG2(800, 4000)	11.9419	2.7211	39.1529	8.796E-04	1.422E-04	12.04%
BMC(150, 600)	2.7381	3.9077	41.8151	7.634E-04	4.996E-04	23.66%

Table 5.2: 99.9% VaR comparison: running time, mean of $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\}$, variance and relative error for the four methods tested for $S = 30$ and $N = 2500$. Computation is broken into two stages: overhead and importance sampling. The true value of $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\}$ is about 1.00E-03. The true value of the integral in equation (1.13) is about 8.515E-04.

we break the running time into the overheads and iteration time, both of which are in seconds. The fourth column lists the total time, which is the sum of overheads (in the second column) and the time for 10 MC iterations (i.e., 10 times of the third column). For SG1(m,n) and SG2(m,n), time consumed in learning m samples is in the second column while time consumed in n samples from the importance sampler is in the third column. GLM*(m,n), on the other hand, has the same overhead for all the three choices of m,n due to the one-time optimization we used. Needless to say, the average iteration time for GLM(m,n) is larger than the sum of the average overhead time and average iteration time for GLM*(m,n). This is because, for GLM(m,n), the loss level associated with the optimization problem is high and the maximum number of steps is reached, without the optimum successfully computed; while for GLM*(m,n), the optimization problem is solved within the time allocated for this step. Lastly, we use the hybrid BMC as discussed in Section 3.5. Because there's no need to solve the importance sampler accurately, as is needed for SG1 and SG2, BMC(m,n) uses a small m that is unchanged for the three different n. We choose m and n for SG$k(m,n)$, BMC(m,n) and GLM*(m,n) such that, if we solve the tail probability just once, i.e., the number of iterations is 1, the total running time would be no larger than for the two crude Monte Carlo approaches. For the four methods SG1(m,n), SG2(m,n), BMC(m,n) and GLM*(m,n), we plot in Figure 5.1 the execution time for one MC iteration, excluding overhead (column two of Table 5.1) versus the error for the computed value.

From the relative errors in Table 5.1, we see that the revised version of GLM, GLM*, performs better than GLM, but it still has a large relative error. The zero variance function estimation approach

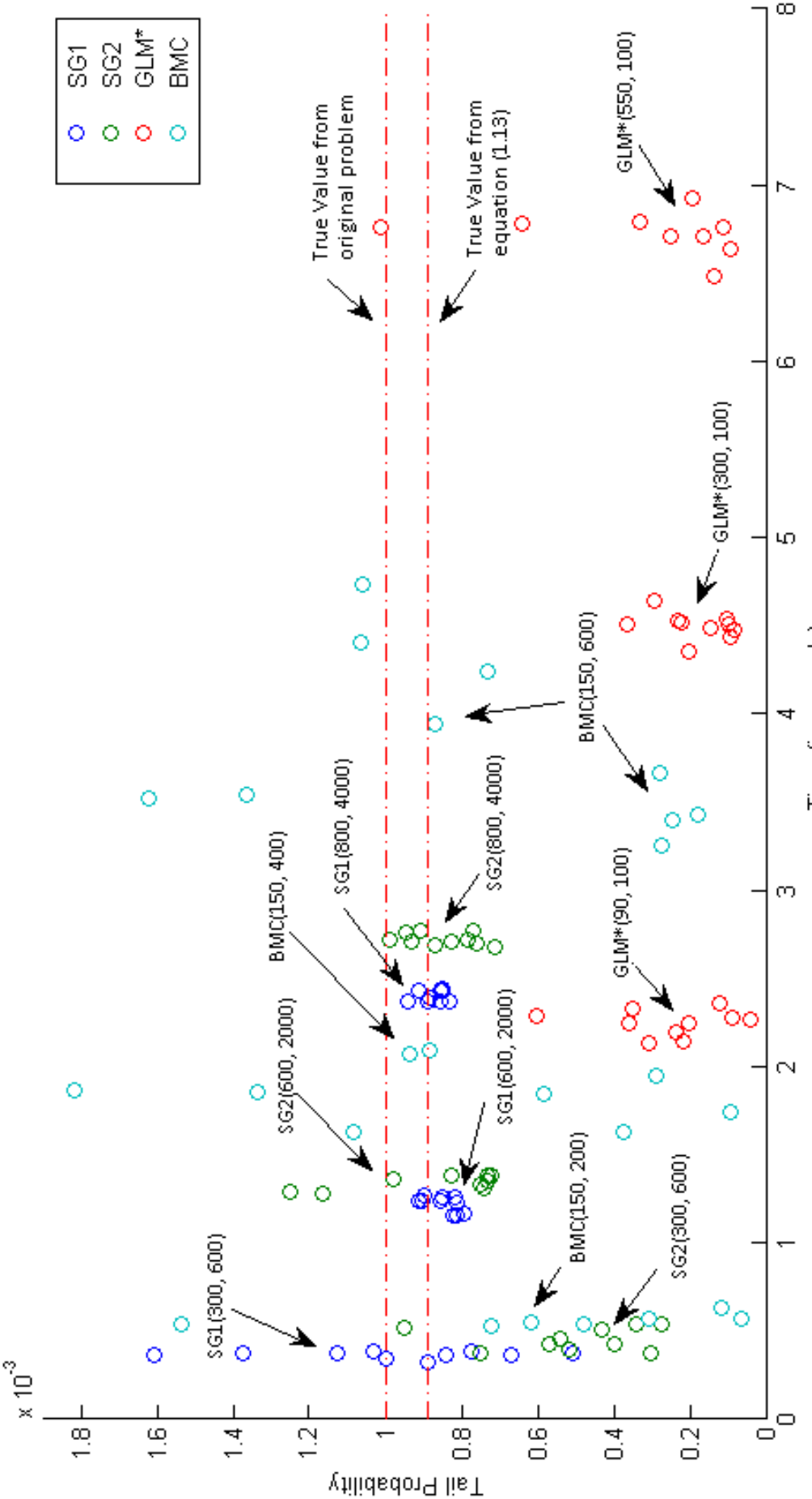


Figure 5.1: 99.9% VaR comparison for $S = 20$ and $N = 2500$.

described in Chapter 2 (both SG1 and SG2) gives an accurate approximation in a short amount of time. We can also visually tell that these two methods reduce variance equally effectively. Although the variance is successfully reduced, note that the blue and green circles in Figure 5.1 are clustered at a level lower than the true value. In fact, they are clustered around the line associated with the true value of the integral on the right side of equation (1.13). The error of SG1 and SG2 is good with respect to equation (1.13), but not so good with respect to $\mathbb{P}\{L_N(\mathcal{Z},\mathcal{E}) > l\}$. Note that BMC is also based on equation (1.13) and thus it is biased as well. For BMC$(150, 400)$, two errors, one from the CLT and the other from the BMC itself, offset each other and give a relatively small total error. However, the errors for BMC$(150, 200)$ and BMC$(150, 600)$ are not nearly as small as the error for BMC$(150, 400)$. Moreover, as we increase the training samples, BMC doesn't achieve good variance reduction. This can also be seen from Figure 5.1, where the cyan circles, indicating the BMC results, don't cluster as well as the corresponding circles for SG1 and SG2 do. For the mean of 7.686E-04 and standard deviation of 5.138E-04, for BMC$(150, 600)$ in Table 5.1, the 95% confidence interval (i.e., $\mu + 1.96\frac{\sigma}{\sqrt{n}}$ ≈1.087E-03 and $\mu - 1.96\frac{\sigma}{\sqrt{n}}$ ≈4.501E-04) is too wide for practical purposes.

We also computed numerical results for $S = 30$. First, note that the overhead for GLM* consists of solving the same optimization problem for all three examples. Thus, we see in Table 5.2 that it is a constant through all the rounds. This is the same for BMC, since we can use the same rough approximation $q(x)$ for all the rounds, as we described in Section 3.5. However we increase the training samples for both SG1 and SG2 from 300 to 600, and finally to 800. Therefore, the overhead for both SG1 and SG2 increases. Secondly, GLM* seeks an exponential twisting parameter for each outer sample, while, for SG1 and SG2, each iteration simply generates samples from a multivariate Gaussian as an importance sampler. Therefore significant computational savings can be seen per iteration for SG1 and SG2 compared to GLM*. More importantly, the numerical results reported in Table 5.2 show that, SG1 and SG2 enjoy a significant decrease in variance as the number of training samples as well as the number of samples from the importance sampler are increased. This conclusion is also supported by the results in Figure 5.2 where we see the blue and green circles, representing SG1 and SG2, respectively, are clustered much closer to the dashed line compared to the red circles, representing GLM*, indicating a better variance reduction for SG1 and SG2 than for GLM*. BMC, on the other hand, shows a similar result as in the previous test for $S = 20$: the circles scatter around. Although the mean is close to the true value and the relative error is therefore small, the variance remains large when we increase the number of training samples.

Note that the two crude Monte Carlo approaches, 2LvlMC and CrudeCLT are not on Figures 5.1 and 5.2, but numerical results for these methods are presented in Tables 5.1 and 5.2. The reason for this is two fold. Firstly, the x-axis of the figures is the average iteration time for each method. If we look at the third column of Tables 5.1 and 5.2, GLM*, SG1, SG2 and BMC are in a similar range, while 2LvlMC and CrudeCLT have much larger average iteration time. More importantly, we focus more on the performance when each approach is applied in the bisection solver. Therefore, we exclude 2LvlMC and CrudeMC from the figures.

5.3 Tail probability calculation

For the numerical results reported in this section, the test problem uses the same digital-credit-states configuration described in Section 5.1. We focus on the tail probability calculation from a less stressed

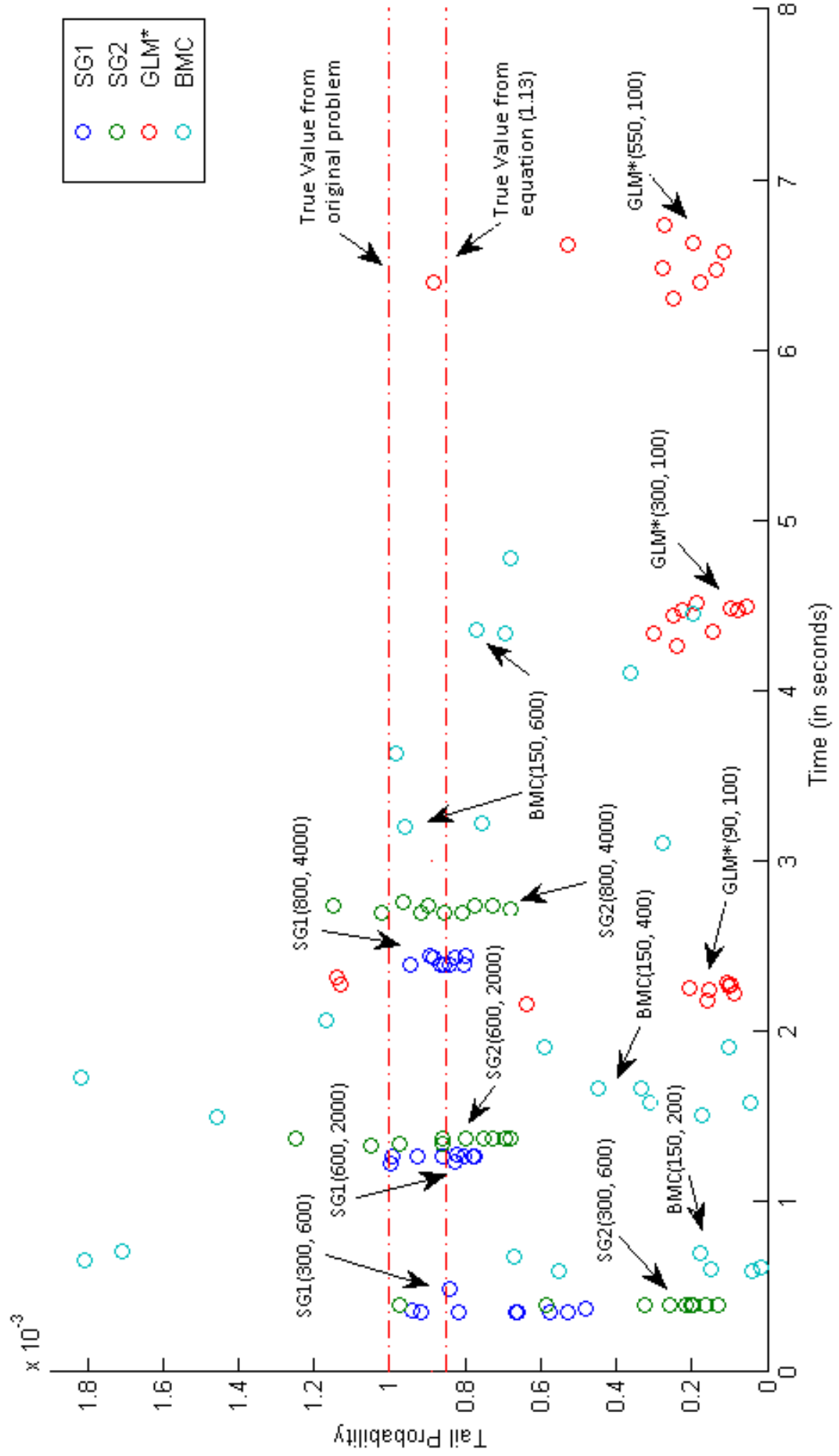


Figure 5.2: 99.9% VaR comparison for $S = 30$ and $N = 2500$.

situation at 90% VaR to a very stressed situation at 99.99% VaR. We also test the calculation for problems under various dimensions S.

For all the numerical methods, we use the same notation as in the previous numerical experiment. We fix m, n and examine different loss levels. Among the three sets we have seen (i.e., SG$k(300, 600)$, SG$k(600, 2000)$ and SG$k(800, 4000)$), SG$k(600, 2000)$ has already shown a significant variance reduction, as the circles are clustered well enough in Figures 5.1 and 5.2. Hence, we choose $m = 600$ as the number of training MCMC samples. On the other hand, we choose $n = 1000$, simply for the reason that, based on a good importance sampler, we can make the SGk approach more competitive from the speed point of view. Thus, we use SG1$(600, 1000)$ and SG2$(600, 1000)$ in this experiment. Having chosen these two methods, we choose GLM*$(200, 100)$, BMC$(300, 500)$, 2LvlMC$(250, 150)$ an CrudeCLT(5000) as representatives of the other methods, since either their computing time or their accuracy is comparable and therefore, we can make comparisons easily between the methods.

Figure 5.3 shows the tail probability for several loss levels for dimension $S = 5$. Points on the graph correspond to loss level $l = VaR$ for the VaR computed by our accurate benchmark two-level MC method at the 90%, 95%, 99%, 99.5%, 99.9%, 99.95%, 99.99% levels. The true value (computed by our accurate benchmark two-level crude MC method 2LvlMC$(5000, 2500)$) is plotted as a solid blue line, while the results for all the other methods are plotted as dashed lines. Note that SG1 is under the blue curves, hence it may not be easily observed in the figure. We also plot the tail probability for the other three different S tested in Figures 5.4 to 5.6.

The mean and variance of the 10 iterations for this test, together with the relative error percentages, are listed in Table 5.3. Firstly, we see that, compared to the crude Monte Carlo approaches, 2LvlMC(250, 150) and CrudeCLT(5000), all the other four approaches indeed achieve a variance reduction. Moreover, in most cases, our new approaches SG1, SG2 and BMC produce results which are much closer to the true value than the results produced by GLM*. This is true for all the four dimensions $S = 5$, $S = 10$, $S = 20$ and $S = 30$ we tested. When we look at SG1, SG2 and BMC for all four values of S tested, SG1 and SG2 reduce the variance significantly more than BMC does, particularly for larger S. Moreover, SG1, in general, performs better in this respect than SG2 does. This may be due to model over-fitting, as at high loss levels, the risk factor shape looks like a Gaussian with a single mode, as explained in Glasserman and Li [3].

We also compare the running time of all the methods. Running times are included in Table 5.4 for $S = 5$, Table 5.6 for $S = 10$, Table 5.8 for $S = 20$ and Table 5.10 for $S = 30$. Similar to the previous testing, the total time is the sum of the overhead for the zero variance function estimation and computing time for 10 iterations. Throughout the four testing dimensions, we see that all methods except GLM* are very stable in terms of running times. Again, this is because our new methods SG1, SG2 and BMC, together with the crude Monte Carlo simulations, are time insensitive to the dimension, while GLM* must exploit the systemic risk surface by solving an optimization problem. We intentionally choose BMC$(300, 500)$ and CrudeCLT(5000) so that the running time of these methods is similar to that of SG1$(600, 1000)$ and SG2$(600, 1000)$, and therefore, the advantage for SGk is obvious: SGk has the least relative error percentage with the smallest variance. BMC, on the other hand, requires further improvement, as its variance is not reduced as much as expected for the large dimension cases, compared to SGk running in a similar amount of time.

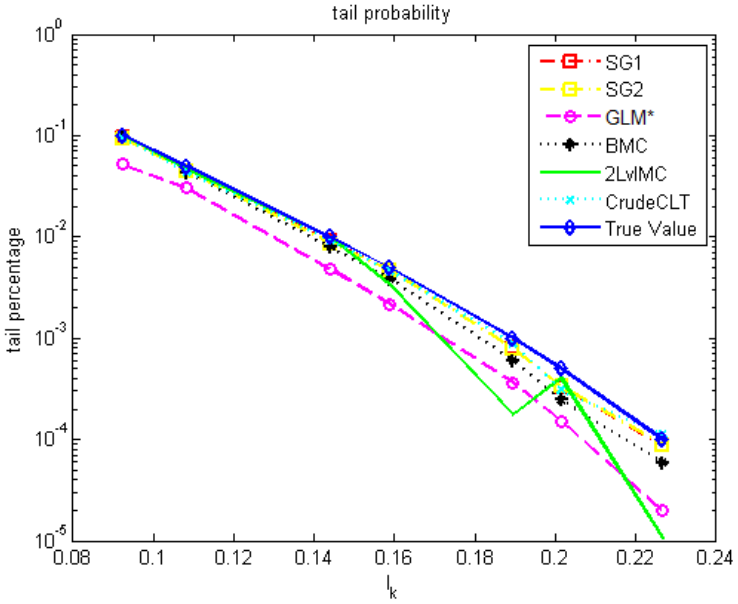


Figure 5.3: tail probability given the loss level l_k for dimension $S = 5$

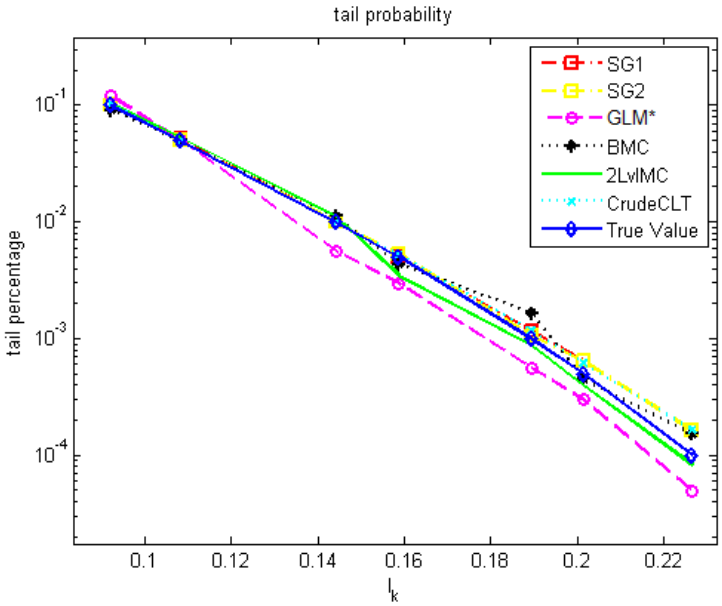


Figure 5.4: tail probability given the loss level l_k for dimension $S = 10$

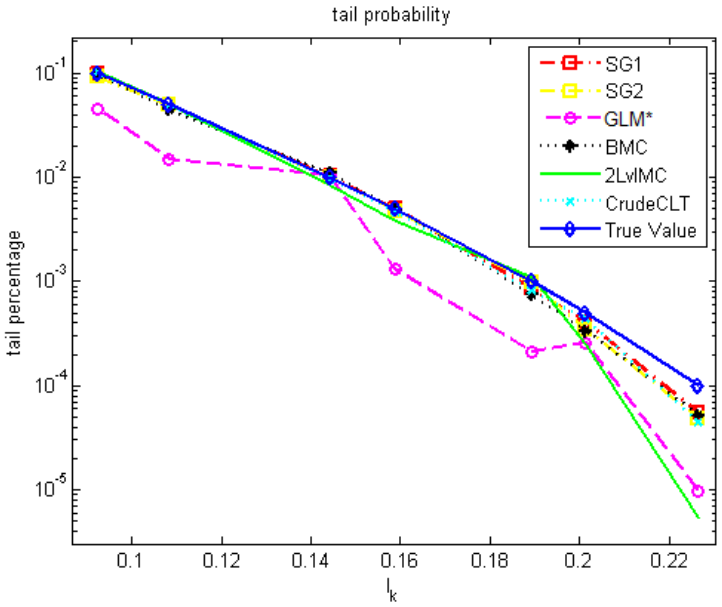


Figure 5.5: tail probability given the loss level l_k for dimension $S = 20$

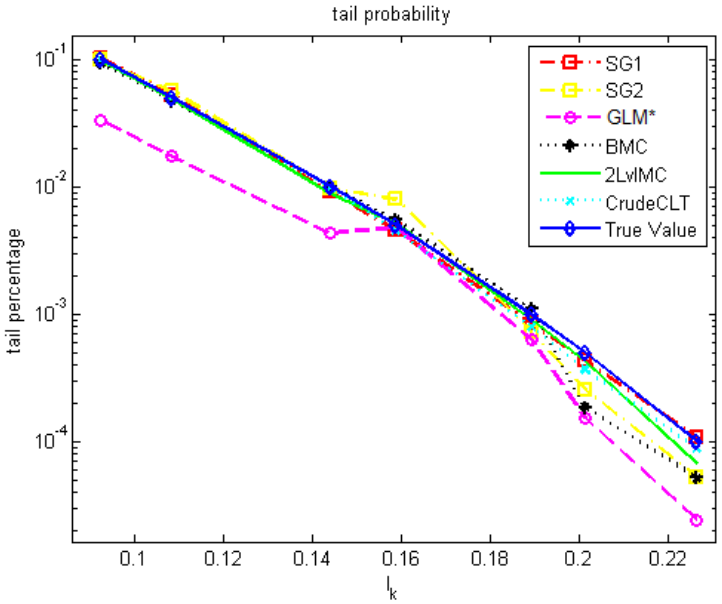


Figure 5.6: tail probability given the loss level l_k for dimension $S = 30$

	SG1(600, 1000)			SG2(600, 1000)			GLM*(200, 100)		
True Val	mean	variance	Rel Err	mean	variance	Rel Err	mean	variance	Rel Err
9.975E-02	9.529E-02	2.407E-05	4.47%	9.404E-02	7.698E-06	5.73%	5.231E-02	7.835E-04	47.56%
4.981E-02	4.492E-02	4.745E-06	9.80%	4.479E-02	2.177E-06	10.06%	3.040E-02	6.414E-04	38.97%
9.985E-03	9.085E-03	9.790E-08	9.02%	8.641E-03	7.948E-08	13.46%	4.760E-03	1.832E-05	52.33%
4.976E-03	4.714E-03	9.835E-08	5.27%	4.735E-03	4.921E-08	4.84%	2.145E-03	1.161E-06	56.90%
9.993E-04	7.990E-04	1.388E-09	20.04%	7.861E-04	7.072E-10	21.34%	3.600E-04	1.527E-07	63.98%
4.999E-04	3.312E-04	3.184E-10	33.75%	3.335E-04	3.842E-10	33.28%	1.509E-04	6.751E-08	69.81%
9.952E-05	8.703E-05	3.632E-11	12.55%	8.831E-05	5.833E-11	11.26%	1.969E-05	3.218E-10	80.22%
	BMC(300, 500)			2LvlMC(250, 150)			CrudeCLT(5000)		
True Val	mean	variance	Rel Err	mean	variance	Rel Err	mean	variance	Rel Err
9.975E-02	9.665E-02	4.186E-05	3.11%	1.001E-01	1.531E-04	0.36%	9.445E-02	4.327E-06	5.32%
4.981E-02	4.354E-02	1.273E-05	12.58%	4.601E-02	7.322E-05	7.62%	4.406E-02	6.635E-06	11.55%
9.985E-03	7.945E-03	3.729E-07	20.43%	9.832E-03	2.648E-05	1.53%	9.244E-03	3.773E-06	7.42%
4.976E-03	3.811E-03	3.727E-07	23.42%	3.331E-03	7.694E-06	33.07%	4.613E-03	5.018E-07	7.29%
9.993E-04	6.063E-04	1.582E-08	39.32%	1.760E-04	7.731E-08	82.39%	8.694E-04	7.040E-08	13.00%
4.999E-04	2.451E-04	6.389E-10	50.98%	4.053E-04	1.526E-06	18.92%	3.036E-04	2.979E-08	39.28%
9.952E-05	5.923E-05	1.164E-10	40.48%	1.067E-05	1.138E-09	89.28%	1.130E-04	1.661E-08	13.52%

Table 5.3: Comparison of the numerical results for $S = 5$

	SG1(600, 1000)			SG2(600, 1000)			GLM*(200, 100)		
	overhead	avg. iter	total	overhead	avg. iter	total	overhead	avg. iter	total
90% VaR	7.3206	0.6505	13.8253	7.5238	0.7244	14.7674	4.2382	3.2108	36.3462
95% VaR	7.3122	0.6546	13.8584	7.5269	0.7199	14.7264	4.2382	3.8245	42.4833
99% VaR	7.3086	0.6531	13.8399	7.4927	0.7165	14.6579	4.2382	4.6320	50.5587
99.5% VaR	7.3054	0.6502	13.8070	7.4999	0.7234	14.7337	4.2382	5.1148	55.3864
99.9% VaR	7.3067	0.6536	13.8426	7.5574	0.7206	14.7631	4.2382	5.5627	59.8652
99.95% VaR	7.3064	0.5897	13.2037	7.4913	0.7197	14.6885	4.2382	5.6217	60.4550
99.99% VaR	7.3050	0.6485	13.7905	7.4745	0.7200	14.6750	4.2382	5.0858	55.0965
	BMC(300, 500)			2LvlMC(250, 150)			CrudeCLT(5000)		
	overhead	avg. iter	total	overhead	avg. iter	total	overhead	avg. iter	total
90% VaR	3.1615	1.6516	19.6776	0.0000	3.0492	30.4921	0.0000	1.5727	15.7271
95% VaR	3.1445	1.6490	19.6343	0.0000	3.0467	30.4668	0.0000	1.5642	15.6421
99% VaR	3.1608	1.5997	19.1576	0.0000	3.0425	30.4250	0.0000	1.5700	15.7000
99.5% VaR	3.2170	1.7954	21.1707	0.0000	3.0398	30.3981	0.0000	1.5710	15.7102
99.9% VaR	3.1908	1.6623	19.8140	0.0000	3.0419	30.4195	0.0000	1.5729	15.7287
99.95% VaR	3.1407	1.5372	18.5126	0.0000	3.0442	30.4415	0.0000	1.5723	15.7226
99.99% VaR	3.1489	1.5334	18.4832	0.0000	3.0433	30.4329	0.0000	1.5704	15.7037

Table 5.4: Running time of the methods tested for $S = 5$. The total time is the sum of the overhead and running 10 iterations. All these times are in seconds.

	SG1(600, 1000)			SG2(600, 1000)			GLM*(200, 100)		
True Val	mean	variance	Rel Err	mean	variance	Rel Err	mean	variance	Rel Err
9.998E-02	1.004E-01	2.624E-05	0.38%	1.004E-01	1.929E-05	0.38%	1.203E-01	2.599E-02	20.34%
5.000E-02	5.117E-02	2.893E-06	2.35%	4.960E-02	4.354E-06	0.79%	5.038E-02	2.389E-03	0.75%
9.997E-03	1.018E-02	1.380E-07	1.82%	1.012E-02	1.592E-07	1.26%	5.614E-03	1.205E-06	43.84%
4.979E-03	5.035E-03	1.422E-08	1.12%	5.261E-03	6.817E-08	5.65%	2.988E-03	4.236E-06	39.99%
9.930E-04	1.170E-03	2.961E-09	17.84%	1.061E-03	3.711E-09	6.86%	5.596E-04	4.053E-08	43.65%
4.974E-04	6.404E-04	4.167E-10	28.73%	6.384E-04	8.253E-10	28.33%	2.996E-04	1.352E-07	39.76%
9.960E-05	1.614E-04	8.487E-11	62.04%	1.622E-04	1.401E-10	62.85%	4.973E-05	4.423E-09	50.07%
	BMC(300, 500)			2LvlMC(250, 150)			CrudeCLT(5000)		
True Val	mean	variance	Rel Err	mean	variance	Rel Err	mean	variance	Rel Err
9.998E-02	9.113E-02	2.952E-05	8.85%	1.023E-01	8.358E-05	2.32%	1.025E-01	2.319E-06	2.51%
5.000E-02	5.003E-02	7.655E-05	0.06%	5.321E-02	6.431E-05	6.42%	4.960E-02	1.445E-06	0.80%
9.997E-03	1.115E-02	1.285E-05	11.56%	1.100E-02	9.805E-06	10.04%	9.965E-03	7.065E-07	0.32%
4.979E-03	4.358E-03	1.146E-06	12.47%	3.493E-03	1.365E-06	29.84%	5.200E-03	2.345E-07	4.44%
9.930E-04	1.656E-03	9.300E-07	66.81%	8.773E-04	5.490E-07	11.65%	1.171E-03	4.044E-08	17.92%
4.974E-04	4.516E-04	2.680E-07	9.22%	3.947E-04	1.235E-07	20.66%	6.166E-04	2.021E-08	23.95%
9.960E-05	1.528E-04	2.953E-08	53.45%	8.267E-05	1.129E-08	17.00%	1.636E-04	1.016E-09	64.24%

Table 5.5: Comparison of the numerical results for $S = 10$

	SG1(600, 1000)			SG2(600, 1000)			GLM*(200, 100)		
	overhead	avg. iter	total	overhead	avg. iter	total	overhead	avg. iter	total
90% VaR	7.3206	0.6505	13.8253	7.5238	0.7244	14.7674	4.2382	3.2108	36.3462
95% VaR	7.3122	0.6546	13.8584	7.5269	0.7199	14.7264	4.2382	3.8245	42.4833
99% VaR	7.3086	0.6531	13.8399	7.4927	0.7165	14.6579	4.2382	4.6320	50.5587
99.5% VaR	7.3054	0.6502	13.8070	7.4999	0.7234	14.7337	4.2382	5.1148	55.3864
99.9% VaR	7.3067	0.6536	13.8426	7.5574	0.7206	14.7631	4.2382	5.5627	59.8652
99.95% VaR	7.3064	0.5897	13.2037	7.4913	0.7197	14.6885	4.2382	5.6217	60.4550
99.99% VaR	7.3050	0.6485	13.7905	7.4745	0.7200	14.6750	4.2382	5.0858	55.0965
	BMC(300, 500)			2LvlMC(250, 150)			CrudeCLT(5000)		
	overhead	avg. iter	total	overhead	avg. iter	total	overhead	avg. iter	total
90% VaR	3.1615	1.6516	19.6776	0.0000	3.0492	30.4921	0.0000	1.5727	15.7271
95% VaR	3.1445	1.6490	19.6343	0.0000	3.0467	30.4668	0.0000	1.5642	15.6421
99% VaR	3.1608	1.5997	19.1576	0.0000	3.0425	30.4250	0.0000	1.5700	15.7000
99.5% VaR	3.2170	1.7954	21.1707	0.0000	3.0398	30.3981	0.0000	1.5710	15.7102
99.9% VaR	3.1908	1.6623	19.8140	0.0000	3.0419	30.4195	0.0000	1.5729	15.7287
99.95% VaR	3.1407	1.5372	18.5126	0.0000	3.0442	30.4415	0.0000	1.5723	15.7226
99.99% VaR	3.1489	1.5334	18.4832	0.0000	3.0433	30.4329	0.0000	1.5704	15.7037

Table 5.6: Running time of the methods tested for $S = 10$. The total time is the sum of the overhead and running 10 iterations. All these times are in seconds.

	SG1(600, 1000)			SG2(600, 1000)			GLM*(200, 100)		
True Val	mean	variance	Rel Err	mean	variance	Rel Err	mean	variance	Rel Err
9.957E-02	9.952E-02	1.675E-04	0.05%	9.158E-02	5.859E-04	8.02%	4.486E-02	5.061E-04	54.94%
4.986E-02	4.950E-02	3.833E-05	0.70%	4.944E-02	2.387E-04	0.83%	1.488E-02	6.520E-05	70.15%
9.901E-03	1.020E-02	1.273E-06	3.04%	9.946E-03	6.689E-06	0.45%	1.046E-02	3.049E-04	5.66%
4.964E-03	5.005E-03	8.781E-07	0.83%	4.718E-03	7.956E-07	4.96%	1.307E-03	6.669E-07	73.66%
9.998E-04	8.484E-04	2.777E-09	15.14%	9.629E-04	1.687E-07	3.69%	2.106E-04	1.735E-08	78.93%
4.949E-04	4.052E-04	4.943E-09	18.13%	3.488E-04	5.958E-09	29.51%	2.580E-04	2.295E-07	47.87%
9.928E-05	5.464E-05	1.020E-10	44.97%	4.876E-05	2.058E-10	50.89%	9.728E-06	5.231E-11	90.20%
	BMC(300, 500)			2LvlMC(250, 150)			CrudeCLT(5000)		
True Val	mean	variance	Rel Err	mean	variance	Rel Err	mean	variance	Rel Err
9.957E-02	9.612E-02	8.262E-05	3.47%	1.021E-01	9.275E-05	2.56%	1.002E-01	5.083E-06	0.66%
4.986E-02	4.460E-02	1.986E-05	10.55%	5.084E-02	7.434E-05	1.97%	4.926E-02	4.160E-06	1.20%
9.901E-03	1.098E-02	5.511E-06	10.94%	8.280E-03	1.068E-05	16.38%	9.615E-03	1.766E-07	2.90%
4.964E-03	4.928E-03	2.170E-06	0.73%	3.856E-03	2.298E-06	22.32%	4.584E-03	1.905E-07	7.65%
9.998E-04	7.423E-04	4.980E-07	25.76%	1.085E-03	1.023E-06	8.55%	8.146E-04	1.973E-08	18.53%
4.949E-04	3.332E-04	8.414E-08	32.66%	2.453E-04	9.241E-08	50.43%	4.346E-04	6.365E-09	12.18%
9.928E-05	5.152E-05	8.718E-09	48.11%	5.333E-06	2.844E-10	94.63%	4.423E-05	3.003E-10	55.45%

Table 5.7: Comparison of the numerical results for $S = 20$

	SG1(600, 1000)			SG2(600, 1000)			GLM*(200, 100)		
	overhead	avg. iter	total	overhead	avg. iter	total	overhead	avg. iter	total
90% VaR	8.8459	0.6348	15.1938	8.9725	0.7051	16.0237	12.4543	3.0033	42.4876
95% VaR	8.8879	0.6093	14.9813	8.7715	0.6965	15.7367	12.4543	3.4515	46.9693
99% VaR	8.9044	0.6025	14.9299	8.7727	0.6937	15.7101	12.4543	4.4081	56.5351
99.5% VaR	8.7832	0.6151	14.9346	8.7824	0.6891	15.6737	12.4543	4.5372	57.8262
99.9% VaR	8.8997	0.6124	15.0232	8.7654	0.6945	15.7107	12.4543	5.1190	63.6439
99.95% VaR	8.7522	0.6124	14.8761	8.8302	0.6940	15.7699	12.4543	5.1898	64.3527
99.99% VaR	8.9072	0.6065	14.9719	8.7633	0.7011	15.7744	12.4543	5.3851	66.3055
	BMC(300, 500)			2LvlMC(250, 150)			CrudeCLT(5000)		
	overhead	avg. iter	total	overhead	avg. iter	total	overhead	avg. iter	total
90% VaR	4.3463	1.7428	21.7744	0.0000	3.1480	31.4803	0.0000	1.5464	15.4643
95% VaR	4.3628	1.5837	20.2001	0.0000	3.1410	31.4103	0.0000	1.5457	15.4566
99% VaR	4.1747	1.5794	19.9689	0.0000	3.1475	31.4753	0.0000	1.5467	15.4672
99.5% VaR	4.3871	1.5703	20.0896	0.0000	3.1397	31.3970	0.0000	1.5568	15.5675
99.9% VaR	4.1557	1.5695	19.8506	0.0000	3.1420	31.4202	0.0000	1.5485	15.4853
99.95% VaR	4.3107	1.5695	20.0055	0.0000	3.1378	31.3777	0.0000	1.5486	15.4858
99.99% VaR	4.3638	1.5889	20.2529	0.0000	3.1416	31.4163	0.0000	1.5477	15.4773

Table 5.8: Running time of the methods tested for $S = 20$. The total time is the sum of the overhead and running 10 iterations. All these times are in seconds.

	SG1(600, 1000)			SG2(600, 1000)			GLM*(200, 100)		
True Val	mean	variance	Rel Err	mean	variance	Rel Err	mean	variance	Rel Err
9.926E-02	1.027E-01	4.316E-04	3.50%	9.917E-02	7.977E-04	0.09%	3.328E-02	1.882E-04	66.47%
4.976E-02	5.258E-02	3.316E-05	5.66%	5.615E-02	3.932E-04	12.84%	1.731E-02	4.995E-05	65.22%
9.955E-03	9.042E-03	4.741E-07	9.17%	9.663E-03	9.730E-06	2.93%	4.306E-03	6.912E-06	56.74%
4.984E-03	4.537E-03	3.983E-07	8.98%	7.988E-03	1.129E-04	60.25%	4.801E-03	2.235E-05	3.67%
9.899E-04	8.429E-04	1.516E-08	14.85%	7.176E-04	2.699E-07	27.51%	6.255E-04	1.135E-06	36.82%
4.954E-04	4.307E-04	9.367E-09	13.08%	2.564E-04	7.328E-09	48.25%	1.532E-04	2.390E-08	69.08%
9.912E-05	1.078E-04	1.968E-10	8.71%	5.244E-05	1.023E-09	47.10%	2.414E-05	3.013E-10	75.65%
	BMC(300, 500)			2LvlMC(250, 150)			CrudeCLT(5000)		
True Val	mean	variance	Rel Err	mean	variance	Rel Err	mean	variance	Rel Err
9.926E-02	9.299E-02	1.151E-04	6.32%	9.777E-02	8.411E-05	1.50%	9.887E-02	7.562E-06	0.40%
4.976E-02	4.741E-02	1.028E-04	4.72%	4.887E-02	2.884E-05	1.78%	4.959E-02	1.231E-06	0.34%
9.955E-03	1.004E-02	6.833E-06	0.88%	8.736E-03	9.850E-06	12.24%	9.651E-03	1.608E-07	3.05%
4.984E-03	5.489E-03	5.635E-06	10.13%	5.085E-03	7.909E-06	2.02%	4.502E-03	1.229E-07	9.68%
9.899E-04	1.092E-03	5.293E-07	10.26%	8.987E-04	6.269E-07	9.22%	7.903E-04	1.400E-08	20.17%
4.954E-04	1.847E-04	1.056E-07	62.73%	4.213E-04	1.264E-07	14.96%	3.667E-04	7.682E-09	25.99%
9.912E-05	5.176E-05	1.033E-08	47.78%	6.667E-05	1.110E-08	32.74%	8.717E-05	3.015E-09	12.06%

Table 5.9: Comparison of the numerical results for $S = 30$

	SG1(600, 1000)			SG2(600, 1000)			GLM*(200, 100)		
	overhead	avg. iter	total	overhead	avg. iter	total	overhead	avg. iter	total
90% VaR	9.5168	0.6155	15.6714	9.3428	0.6840	16.1827	23.3733	3.5252	58.6256
95% VaR	9.3080	0.6051	15.3588	9.3752	0.6878	16.2532	23.3733	3.8672	62.0458
99% VaR	9.4624	0.5905	15.3675	9.5073	0.6876	16.3836	23.3733	4.6442	69.8158
99.5% VaR	9.2975	0.5961	15.2588	9.3924	0.6860	16.2521	23.3733	4.9409	72.7823
99.9% VaR	9.3197	0.6052	15.3713	9.5709	0.6829	16.3994	23.3733	5.3062	76.4349
99.95% VaR	9.4685	0.6041	15.5092	9.3032	0.6863	16.1658	23.3733	5.4054	77.4275
99.99% VaR	9.4267	0.6060	15.4870	9.4830	0.6859	16.3420	23.3733	5.5266	78.6395
	BMC(300, 500)			2LvlMC(250, 150)			CrudeCLT(5000)		
	overhead	avg. iter	total	overhead	avg. iter	total	overhead	avg. iter	total
90% VaR	4.7210	1.6487	21.2084	0.0000	3.0888	30.8875	0.0000	1.5788	15.7876
95% VaR	4.8641	1.5923	20.7874	0.0000	3.0924	30.9241	0.0000	1.5796	15.7959
99% VaR	4.6649	1.6043	20.7076	0.0000	3.0873	30.8726	0.0000	1.5840	15.8403
99.5% VaR	4.7078	1.5705	20.4132	0.0000	3.0913	30.9132	0.0000	1.5770	15.7705
99.9% VaR	4.6592	1.6393	21.0525	0.0000	3.1137	31.1368	0.0000	1.5786	15.7863
99.95% VaR	4.8684	1.6448	21.3163	0.0000	3.0961	30.9607	0.0000	1.5787	15.7870
99.99% VaR	4.7544	1.6223	20.9772	0.0000	3.0959	30.9587	0.0000	1.5807	15.8071

Table 5.10: Running time of the methods tested for $S = 30$. The total time is the sum of the overhead and running 10 iterations. All these times are in seconds.

Chapter 6

Conclusion

In this research paper, we used the Central Limit Theorem to approximate portfolio credit risk and then presented two novel approaches motivated from machine learning to evaluate the approximation. The first method adopts an importance sampling technique based on an estimate of the zero variance function computed using Markov Chain Monte Carlo and a Gaussian Mixture Model. The second method makes use of the Bayesian Monte Carlo approach to model the integrand as a Gaussian Process and treats the integral as a Bayesian inference problem. We compare our two new methods to GLM, a method proposed by Glasserman and Li [3]. Numerical results show that our two new approaches achieve significant improvement compared to GLM. However, our current implementation of the first approach is more effective at reducing the variance than our current implementation of the second approach. Future work will focus on improving the Bayesian Monte Carlo method by exploring better ways to model the integrand function.

Bibliography

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

[2] Joshua C.C. Chan and Dirk P. Kroese. Improved cross-entropy method for estimation. *Statistics and Computing*, 22(5):1031–1040, 2012.

[3] Paul Glasserman and Jingyi Li. Importance sampling for portfolio credit risk. *Management Science*, 51(11):1643–1656, 2005.

[4] Michael B. Gordy. A risk-factor model foundation for ratings-based bank capital rules. *Journal of Financial Intermediation*, 12:199–232, December 2003.

[5] Meng Han. Approximations and Optimization for Portfolio Credit Risk in the Gaussian Copula Factor Model. *Research Proposal*, May 2011.

[6] Meng Han. Loss distribution and optimization based on the structural model for credit portfolios. *University of Toronto Research Paper*, 2011.

[7] Robert C. Merton. On the pricing of corporate debt: The risk structure of interest rates. *Journal of Finance*, 29:449–470, 1973.

[8] Michael Osborne, Roman Garnett, Zoubin Ghahramani, David K. Duvenaud, Stephen J. Roberts, and Carl E. Rasmussen. Active learning of model evidence using Bayesian quadrature. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, pages 46–54, 2012.

[9] Radford Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6:353–366, 1996.

[10] Radford Neal. Slice sampling. *Annals of Statistics*, 31(3):705–767, 2003.

[11] Anthony O'Hagan. Bayes-Hermite Quadrature. *Journal of Statistical Planning and Inference*, 29(3):245–260, 1991.

[12] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. http://www.gaussianprocess.org/gpml/.

[13] Carl Edward Rasmussen and Zoubin Ghahramani. Bayesian Monte Carlo. *Advances in Neural Information Processing Systems*, pages 489–496, 2002.

[14] Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16(4):351–367, 2001.

[15] Jeffrey S. Rosenthal. STA4502 (Monte Carlo Estimation) Lecture Notes. January - Feburary 2013. http://probability.ca/jeff/teaching/1213/sta4502/notes.pdf.

[16] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

[17] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer Science & Business Media, 2004.

[18] Christopher K. I. Williams and David Barber. Bayesian classification with Gaussian processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 20(12):1342–1351, 1998.