Control Variates For Bayesian Computation

Candidate Number: MWWR6

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Master of Science

of

University College London.

Department of Statistical Science

University College London

September 20, 2021

I, Candidate Number: MWWR6, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Markov Chain Monte Carlo (MCMC) techniques, belonging to Bayesian group of computation methods, has potential to sample non-normalised posterior distributions of model parameters, in contrast to the basic Monte Carlo (MC) method, so as to obtain the estimated quantity. The variance of these estimators could then be reduced by employing the control variates (CVs) method. In this thesis, a novel framework that encompasses and generalises the existing zero variance (ZV) approach as one of the control variates (CVs) method is presented by considering a distinct Stein operator. Furthermore, an approach for estimating the coefficients of different cases under this framework is also proposed. In addition, multiple cases involved in the novel framework are experimented with by applying the Bayesian models to different datasets. The results of the experiments are analysed in terms of their respective variance reduction as well as the trade-off between the method and the corresponding computational costs. It shows GZV method could improve ZV method in efficiency. Further, approach to construct intermediate control variates to supplement ZV method is suggested and meaning features are summarized for future extension.

Acknowledgements

My sincere and hearty thanks and appreciations go firstly to my supervisor, Dr François-Xavier Briol, whose suggestions and explanations have given me much insight into these studies. It has been a great privilege and joy to study under his guidance and supervision. Furthermore, if there were no great patience as well as the additional support for me to pass the insomnia time, I would have had no confidence to face this research-type thesis and produce any meaningful content individually.

Next, I would like to express my gratitude to Zhuo Sun, the student of François-Xavier, for the meeting to suggest the problems I met.

Last, but by no means least, my appreciations also go to Yang Chenlu who I treasure much and sincerely wish here that despite years later, nothing changes except for age!

Contents

List of Figures

List of Tables

0.1 General Notation

Most notations presented in this paper are explained as and when they appear, while some general notational conventions used throughout the paper are introduced here at the onset for convenience. Let $\mathbb{R}^n$ and $\mathbb{Z}^n$ denote the n-tuples (vector) of real numbers and integers, respectively, where n is an integer and $n \geq 1$. The n-tuple (vector) of real numbers $(x_1, ..., x_n)^{\mathrm{T}}$ is presented in bold type $\boldsymbol{x}$ for short and x_i means the i-th element of n-tuples (vector) $\boldsymbol{x}$. The vector $\boldsymbol{x}$ is regarded as column vector by default, whereas $\boldsymbol{x}^{\mathrm{T}}$ could be treated as a row vector. When $n = 1$, x is equivalent to $\boldsymbol{x}$. The centered point $\cdot$ is used to designate the dot product of two vectors: $\boldsymbol{x} \cdot \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{x}$ and it also denotes the multiplication between two constants sometimes. Accordingly, $||\boldsymbol{x}|| = (\boldsymbol{x} \cdot \boldsymbol{x})^1/2$ could denote the Euclidean norm on $\mathbb{R}^n$. Matrix, consisted of sequence of (column) vectors $(\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$, is presented in capital form with bold type $\boldsymbol{X}$ where X_{ij} means the element in i-th row and j-th column.

In terms of statistical representation, by default, $p(\boldsymbol{x})$ denotes the probability density function (pdf) of variable $\boldsymbol{X}$, while $P(\boldsymbol{x})$ presents the probability where x is the observed value (sample). As the thesis is within the context of Bayesian computation, θ is set to be the parameter variable by default, which may need to be sampled, estimated, and so forth. Sometimes, $\pi(\theta)$ could also present pdf of the parameter θ and this notation appears commonly in the context of MCMC algorithm as a stationary distribution. Another notation appears in the thesis to describe order k is $O(n^k)$ such that $\frac{O(n^k)}{n^k}$ is a constant when $n \to \infty$.

0.2 Introduction to The Thesis

In terms of problems regarding Bayesian inference, in order to obtain the quantity of interest, one has to face the inherent issues of integration, whereby the exact computation is practically infeasible, especially when the problems are in higher dimensions. To tackle this, the Markov Chain Monte Carlo (MCMC) method, which comprises elements of Monte Carlo (MC), could be applied by providing samples for estimations where the precision depends is contingent upon the sample size. However, sampling is not computationally free and usually takes a significant amount of time or other computational resource. To increase the precision under the same computational cost of direct sampling, the control variates (CVs) method (as a variance reduction technique) has potential to reduce the error of the estimation of the unknown quantity by exploiting closed-form results. In brief, it could identify another function – termed the control variate – whose expectation is zero but could reduce variance through replacing the original function for the estimator by adding up to it.

The main challenge in applying CVs has been in constructing them. Two relatively recent and practical methods are explained in this thesis, and are respectively termed as zero variance and control functionals. For the purposes of this thesis, the former one is the scope of our analytical focus.

Due to the distinct directions of MC (including MCMC) for sampling and CVs for variance reduction, the requisite background is divided in order to facilitate a comprehensive interpretation. The background shall be presented across chapter 1 and chapter 2, respectively. In chapter 1, Bayesian inference problems are introduced in more details and sufficient knowledge underpinning MC (including simple MC and MCMC) for sampling are interpreted. In addition, as MCMC shall be herein applied in practice, the relevant topic regarding implementation (such as post-processing and diagnostic) are also explained.

In chapter 2, the defined 'efficiency' and several other pertinent criteria are interpreted for variance reduction techniques. Then, the underlying premise of CVs is explained in greater detail. Two types of Stein Operator, along with the relationship between them, are illustrated from the start, followed by the two CVs methods: the zero variance (ZV) method and control functionals (CF) method. As ZV is object of analytical focus for this thesis, it is elaborated upon, while CF is merely outlined and considered in relation to ZV.

With regard to the ZV method, even if the construction of a control variate is not based on a Stein operator, it could still be viewed as using one, specifically that which is referred to as the scalar-valued Langevin (SL) Stein operator, in order to support mapping scalar-valued function. By trying the vector type of this operator for mapping vector-valued functions, it is ascertained that control variates with greater flexibility could be produced and this entire approach for construction is defined as generalised zero variance (GZV) method by this thesis, due to its convenience and similarity to the ZV method. This thesis aims to exploit this GZV approach in order to extend the ZV method.

In chapter 3, the novel framework to apply the GZV method is originally proposed, revealing that the current ZV method could be effectively encompassed and generalised. Then, the procedure to optimise the coefficients in the control variate under this framework shall be articulated. Following this, several types of control variate, including ones that could not be constructed by the existed ZV method, are tested across multiple combinations of the Bayesian model applied to different datasets. Finally, focusing on sampling for posterior mean, the results are analysed and discussed by behaviour of variance reduction and the trade-off between variance reduction and computational costs. It shows that one type of control variate could be more efficient than the existed most efficient type constructed by ZV method. Further, the approach to construct intermediate control variates between the mere two existed types (by ZV) is suggested according to results and some meaningful features are summarised for the idea to construct control variates in higher order using GZV method. The limitations of this study, as well as suggestions for future research are considered and presented in chapter 4. In general, apart from improving the efficiency (less variance with identical computational cost), the key contribution of this proposed GZV method is that it could provide a more generalised framework for further investigation to extend the ZV method as part of a polynomial based approach.

Chapter 1

Background on MC MCMC

1.1 Introduction to Monte Carlo methods

1.1.1 Issues about Bayesian computation

Bayesian inference is a method related to statistical inference, which primarily involves applying Bayes' theorem to derive the updated posterior distribution from the pre-decided prior distribution and the likelihood function determined by the statistical model and collected observed data. In formula form, this process is

$$p(\boldsymbol{\theta}|x) = \frac{p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \propto p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{1.1}$$

where $\boldsymbol{\theta}$ and $\boldsymbol{x}$ represents parameters and data in the likelihood function $p(\boldsymbol{x}|\boldsymbol{\theta})$, while $p(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}|\boldsymbol{x})$ are the prior and posterior function for $\boldsymbol{\theta}$,respectively. With regard to Bayesian inference, there are some typically intractable integration problems which need to be analytically solved [1].

1. Normalizing the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{x})$ which is proportional to the multiplication of $p(\boldsymbol{\theta})$ and $p(\boldsymbol{x}|\boldsymbol{\theta})$ according to Bayes' theorem.

2. Calculating the marginal posterior distribution where the parameter variable vector is $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_n)$ and i is in $1, ..., n$:

$$p(\theta_i|\mathbf{x}) = \int p(\theta_1, \ldots, \theta_n|\boldsymbol{x})\, d\theta_1 ... d\theta_{i-1} d\theta_{i+1} ... d\theta_n$$

3. Obtaining the summary statistics, such as the expectation of some form of $\boldsymbol{\theta}$ (which is focused in this thesis) under the condition of $\boldsymbol{x}$:

$$E_{p(\boldsymbol{\theta}|\boldsymbol{x})}[f(\boldsymbol{\theta})] = \int f(\boldsymbol{\theta})\, p(\boldsymbol{\theta}|\boldsymbol{x})\, d\boldsymbol{\theta}$$

Integrating and solving the above problems is extremely troublesome when $\boldsymbol{\theta}$ is high-dimensional, which occurs frequently, such as in Bayesian hierarchical model. In addition, for many models, the posterior distribution lacks the analytical form so that it must be approximated. Since this thesis focuses on 'computation' rather than 'inference', examples of Bayesian inference are not included. Bayesian computation, which constitutes a class of computational methods rooted in Bayesian statistics, is able to tackle these integration problems, with the two main approaches comprising simulated-based methods such as Markov chain Monte Carlo (MCMC) included in Monte Carlo (MC) methods, and approximation methods, such as Variational Bayesian inference. As a set of powerful simulating methods, MC methods (including MCMC) actually catalyse the development of Bayesian statistics [2] and they shall be interpreted next.

1.1.2 Monte Carlo (MC) methods

In essence, Monte Carlo (MC) methods are a broad class of computational algorithms to generate the random objects or processes in order to obtain numerical results where the underlying concept is to use randomness. In order to solve the problems which might be deterministic in principle, Monte Carlo techniques could repeat the experiment many times so that many quantities of interest could be obtained according to concepts in statistical inference like Law of Large Numbers. In general, it could be split into three typical uses in terms of Bayesian statistics: 1. Sampling to gather information of parameter variables in posterior distribution 2. Estimating useful quantities. For MC methods in this thesis, it is considered to use MCMC in order to obtains samples to estimate the posterior mean such that CVs could be implemented to reduce the variance of the estimator.

1.1.3 Advantages of Monte Carlo methods

The reason why MC methods is widely used is due to its several advantages [2]:

1. It is simple, flexible and scalable so that it could be used to widespread models and produce a relative accurate result which may be much more accessible than analytic methods like estimating value of integration problems. In addition, different parts for MC algorithms could be implemented independently which could lead to great improvement of computation efficiency. Moreover, its implementation is based on computer so that it may be more practical as the development of the computer.

2. It could bring the comprehension of the behavior of random systems or data based on the multiple random experiments, which is the essence of statistics, so that mutualistic symbiosis could exist between MC techniques and statistics. Recently, modern statistics increasingly relies on computational tools such as resampling and MCMC when analyzing very large amount and/or high dimensional data sets.

3. It is based on statistical knowledge like central limit theorem such that, for instance, its convergence could be guaranteed and efficiency to converge could be estimated. It's also the fundamental reason why variance reduction could be applied.

After the general interpretation of MC is given, the concrete methods should be introduced since they are implemented in this thesis actually.

1.1.4 Simple Monte Carlo Methods

Ahead of MCMC, simple (basic) MC is interpreted in the beginning to illustrate the fundamental ideas of MC methods (including MCMC) with regard to estimating. Commonly, the quantity of interest estimated and focused by most paper (and this paper) is the expected value $\mu = E_{p(\boldsymbol{\theta})}[f(\boldsymbol{\theta})]$ of $f(\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ ($\boldsymbol{\theta} \in \mathbb{R}^d$) is the parameter variable under the probability density function $p(\boldsymbol{\theta})$ and f is a real-valued function defined over d. For some other paper, for example, they may say that they focus on the approximation of the integral of an arbitrary function with respect to a distribution. As it is said before that in terms of Bayesian computation, the key point is to deal with the intractable integration problems. Accordingly, the purposes of these paper are identical indeed. Here the former aim is chosen in this thesis in order to keep consistency.

From the definition of expected value, it is known that $\mu = \int f(\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$. For basic MC, to estimate μ, sequence of samples $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_n$ are generated independently and randomly from the distribution of $p(\boldsymbol{\theta})$ so that the average could be taken as the estimation. Simply, the above could be written as:

$$\frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{\theta}_i) \approx \mu = \int f(\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta} \tag{1.2}$$

The convergence about the estimation could be proved by the Strong law of large numbers which states (write $\frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{\theta}_i) = \hat{\mu}_n$ and $\mathrm{Var}[f(\boldsymbol{\theta}] = \sigma^2$):

$$P(\lim_{n\to\infty} |\hat{\mu}_n - \mu| = 0) = 1$$

With the condition $E_{p(\boldsymbol{\theta})}[f(\boldsymbol{\theta})^2] < \infty$, according to central limit theorem:

$$\sqrt{n}.(\hat{\mu}_n - \mu) \to N(0, \sigma^2)$$

the estimated statistic $\hat{\mu}_n$ converges to μ at the rate $O(n^{-1/2})$ or simply at $\sqrt{n}$. Although increasing n could increase precision of the estimator, sometimes, $\boldsymbol{\theta}$ and f may be expensive to simulate and evaluate, respectively, or even only a limited number of samples could be obtained. For example, model named Met Office for future climate simulations required the order of 10^6 core hours per simulation [3]. As the error variance of simple MC integration is $\frac{\sigma^2}{n}$, finding a way to reduce σ

is an alternative way to improve precision and such methods are called variance reduction techniques such as CVs.

1.2 Introduction to Markov Chain Monte Carlo (MCMC)

The problem of basic MC is that it could not deal with the non-normalized density distribution, which is necessary for Bayesian computation. As the result, MCMC should be applied and it shall be interpreted. However, the estimation still follows the above structure regardless of some details like ”independent sampling”.

1.2.1 Introduction to Markov Chain

To introduce MCMC, the essentials of Markov chain is explained in the beginning. Markov chain, in brief, is a stochastic model describing a sequence of possible events occurring on continuous time or discrete time where the probability of each event merely relies on the state attained in the last one event. Due to the nature of simulation, the type of Markov chain focused on Bayesian computation here is only discrete-time stochastic process (continuous-time Markov chain is not considered). In fact, it is mentioned by [4] that Markov chain related with MCMC is finte state-space indeed because of the use of pseudo-random generators and the representation of numbers in computer. However, for convenience, it is considered to use general state-space to allow for continuous support distributions rather than use discrete support distributions to approximate these distributions. To introduce Markov chain in this section, the finite as well as the general state-space type are illustrated.

In general, considering the sequence of random variables $\theta_1, ..., \theta_n$, Markov chains are constructed from the transition kernel K that is a conditional probability density where $\theta_n \sim K(\theta_n, \theta_{n+1})$ during the setup of MCMC. For finite state-space Markov chain, K is a transition matrix $\boldsymbol{P}$ and $\boldsymbol{P}_{ij} = P(X_{n+1} = \omega_j | X_n = \omega_i)$ where ω_i, ω_j belongs to the (finite) state space $\Omega = \omega_1, \omega_2, ..., \omega_M$ of size M. Commonly, ¶ remains to be invariant along the sequence and this is called homogeneous. For general-state Markov Chain, a typical example as well as a key role in MCMC algorithms is random walk process that means $X_{n+1} = X_n + \varepsilon_n$ where ε_n is independent from the sequence. Then, another important concept involved in Markov chain is stationary probability distribution that means a distribution $\pi(\theta)$ such that $\theta_{n+1} \sim \pi(\theta)$ if $\theta_n \sim \pi(\theta)$. It entitles the strong stability to the chains satisfied that in MCMC settings. Depending on regularity conditions, the Markov chain could converge in distribution to a unique stationary distribution without subject to the initial state. In formula form, it is

$$\lim_{n\to\infty} P(\theta_n = \omega | \theta_0 = \omega_0) = \pi(\omega)$$

where the initial state ω_0 and the final state ω could be arbitrary in the state space. This convergence is guaranteed by two definitions, irreducible and aperiodic, for finite state-space type, while for the general state-space type, the similar result still holds even if situation becomes more complicated and it needs one more condition that is the existence of stationary distribution. The proof is numerous while a summary could refer to [5]. Briefly, irreducible represents that Kernel K could allow for free walks over the state space and aperiodic means that there is any destined "round trip" for any state with some periods. To avoid the complicated theoretic analysis, two examples for finite/general state-space Markov chain are shown, respectively.

For finite-state markov chains, consider the initial probability distribution for states written in (row) vector form $(0,0,1)$ and the transition matrix $\boldsymbol{P}$ set to be

$$\begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}$$

Then, the probability distribution for states after iterations could be observed (by Figure 1.1) to converge to stationary since the conditions for this mentioned above are satisfied. In addition, it could also be noted that the convergence of (by 1.2) $\boldsymbol{P}^n$

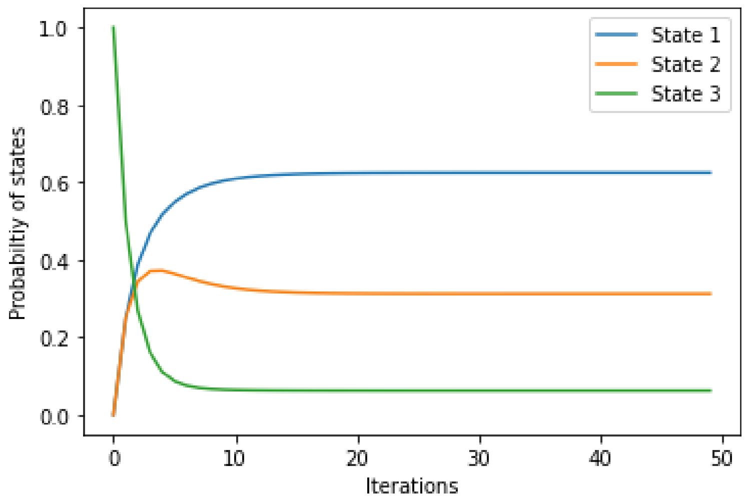

Figure 1.1: Plot showing convergence to stationary distribution (50 iterations)

exists and it is given by

$$\lim_{n\to\infty} \boldsymbol{P}^n = \begin{bmatrix} \pi(\omega_1) & \pi(\omega_2) & \cdots & \pi(\omega_M) \\ \pi(\omega_1) & \pi(\omega_2) & \cdots & \pi(\omega_M) \\ \vdots & \vdots & \ddots & \vdots \\ \pi(\omega_1) & \pi(\omega_2) & \cdots & \pi(\omega_M) \end{bmatrix}$$

For general-state Markov chains, an example is that for i in $1,\ldots,n$, the sequence of

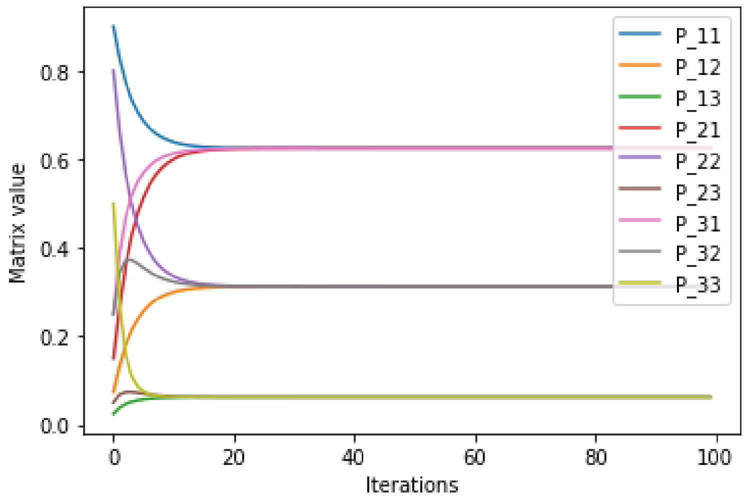

Figure 1.2: Plot showing convergence of n-step transition matrix (50 iterations)

samples θ_i is generated according to

$$\theta_{i+1} = \frac{\theta_i}{2} + \varepsilon_i$$

where $\varepsilon_i \sim N(0,1)$. Then the stationary distribution would be $N(0, \frac{4}{3})$. Here four cases are plotted in Figure 1.3 where the initial state of two figures is -10 and that of the other two is 10.

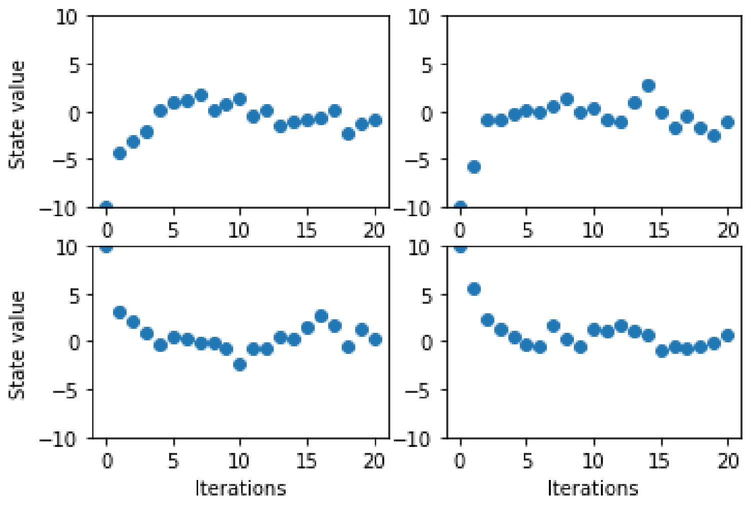

Figure 1.3: Plot showing convergence for general-state Markov chains (20 iterations)

1.2.2 MCMC Using Metropolis-Hasting algorithm

With the essential knowledge of Markov chain, Markov Chain Monte Carlo (MCMC) using the Metropolis-Hasting algorithm can be interpreted as follows. The Metropolis-Hasting algorithm, indeed, is to construct a Markov chain such that the existence of stationary distribution π is guaranteed. This algorithm is named after in [6] authored by Metropolis et al. and extended by Hastings et al. to the more general form in [4].

Regarding the target distribution (for sampling) as the stationary distribution π, generating the samples could be based on the detailed balance condition

$$\pi(\theta)P(\theta'|\theta) = \pi(\theta')P(\theta|\theta') \tag{1.3}$$

where θ and θ' are arbitrary states and P is the conditional distribution that should be constructed. Similar to what is done in acceptance-rejection method, the proposal term $Q(\theta'|\theta)$ is raised, whereas the equation in (1.3) could be guaranteed

$$\pi(\theta)Q(\theta'|\theta) \neq \pi(\theta')Q(\theta|\theta')$$

To solve this problem, it is still the rejection sampling could be referred to. The acceptance term to describe the acceptance probability from state θ to θ' is stated as $A(\theta'|\theta)$ and it could form:

$$\pi(\theta)Q(\theta'|\theta)A(\theta'|\theta) = \pi(\theta')Q(\theta|\theta')A(\theta|\theta') \tag{1.4}$$

where $A(\theta'|\theta)$ and $A(\theta|\theta')$ could be given by

$$A(\theta'|\theta) = \pi(\theta')Q(\theta|\theta') \ \& \ A(\theta|\theta') = \pi(\theta)Q(\theta'|\theta)$$

In addition, it could be noted that if a given acceptance function A could provide the detailed balance and so will A/c for any constant c, while the range of A is limited from 0 to 1 since it is to describe probability. And A as thee acceptance shall be as large as possible for sake of sampling. Therefore, expand two sides of equation (1.4) with the same constant c such that the maximal value between $cA(\theta'|\theta)$ and $cA(\theta|\theta')$ is equal to 1. If $A(\theta'|\theta)$ is larger, then c$A(\theta'|\theta)$ directly equals to 1 and thus the previous $A(\theta'|\theta)$ shall be replaced by 1. Otherwise, since $cA(\theta'|\theta) = \frac{\pi(y)Q(\theta|\theta')cA(\theta|\theta')}{\pi(\theta)Q(\theta'|\theta)} = \frac{\pi(\theta')Q(\theta|\theta')}{\pi(\theta)Q(\theta'|\theta)}$, the new extended $A(\theta'|\theta)$ shall be $\frac{\pi(\theta')Q(\theta|\theta')}{\pi(\theta)Q(\theta'|\theta)}$. The official name of the new formed $A(\theta'|\theta)$ is called Metropolis-Hasting acceptance probability and could directly written as:

$$A(\theta'|\theta) = min\{1, \frac{\pi(\theta')Q(\theta|\theta')}{\pi(\theta)Q(\theta'|\theta)}\}$$

Since the procedure to obtain $A(\theta|\theta')$ is equivalent, it is omitted. This new formed $A(\theta'|\theta)$ could be much larger than it if the above procedures are not implemented. As the result, this Metropolis-Hasting algorithm makes MCMC method become more practical indeed. In addition, to carry out the acceptance straightforwardly, U belongs to $\boldsymbol{U}(0,1)$ could be sampled so that the transition (from a state to another) is only accepted if $U \leq A(\theta'|\theta)$. In summary, the concrete procedures for MCMC sampling based on Metropolis-Hasting algorithm is listed in Algorithm 1.

Algorithm 1 Metropolis-Hasting algorithm

```
1: Initialization: Prepare the initial state $\theta_1$, proposal transition matrix $\boldsymbol{Q}$ and
   identify the target distribution $\pi$ for sampling. Set the number of iterations $n$
   and start the loop from $i$=1
2: while $i \leq n$ do
3:     Sample $\theta'$ from $Q(\theta'|\theta_i)$
4:     Sample U from $\boldsymbol{U}(0,1)$
5:     if $U < A(\theta'|\theta_i) = min\{1, \frac{\pi(\theta')Q(\theta|\theta')}{\pi(\theta)Q(\theta'|\theta)}\}$ (or just $U < \frac{\pi(\theta')Q(\theta|\theta')}{\pi(\theta)Q(\theta'|\theta)}$ for simplify-
       ing) then
6:         return $\theta_{i+1} = \theta'$
7:     else
8:         return $\theta_{i+1} = \theta'$
9:     end if
10: end while
Output: The chain $\theta_1, ..., \theta_n$ could be obtained.
```

In terms of the choice of proposal transition $Q(\theta'|\theta)$, it needs to be noted that detailed balance condition could not guarantee the transition to be irreducible. An extreme but simple case is $Q(\theta|\theta) = 1$ for θ locates on all states. Commonly, the proposal transition could be chosen as normal distribution $\boldsymbol{N}(\mu = \theta, \sigma^2 = 1)$, which is symmetric so that $Q(\theta'|\theta) = Q(\theta|\theta')$. Therefore, the value of Metropolis-Hasting acceptance probability could be directly written as $min\{1, \frac{\pi(\theta')}{\pi(\theta)}\}$. Here is an example where the target distribution is set to be a bimodal one whose pdf is the average of $\boldsymbol{N}(-5,9)$ and $\boldsymbol{N}(5,9)$. Coding with notes and the histogram plot 1.4 for sampled $\theta_1, ..., \theta_n$ are shown in the following: The intermediate red dashed line at value 0 represents mean value of the MCMC samples as well as the median due to overlapping. The two black dotted lines indicate 25 percentile and 75 percentile and they locate at -5 and 5, respectively. The positions of these lines are roughly equal to the corresponding value from the target distribution, which suggests the samples follows the target distribution accurately.

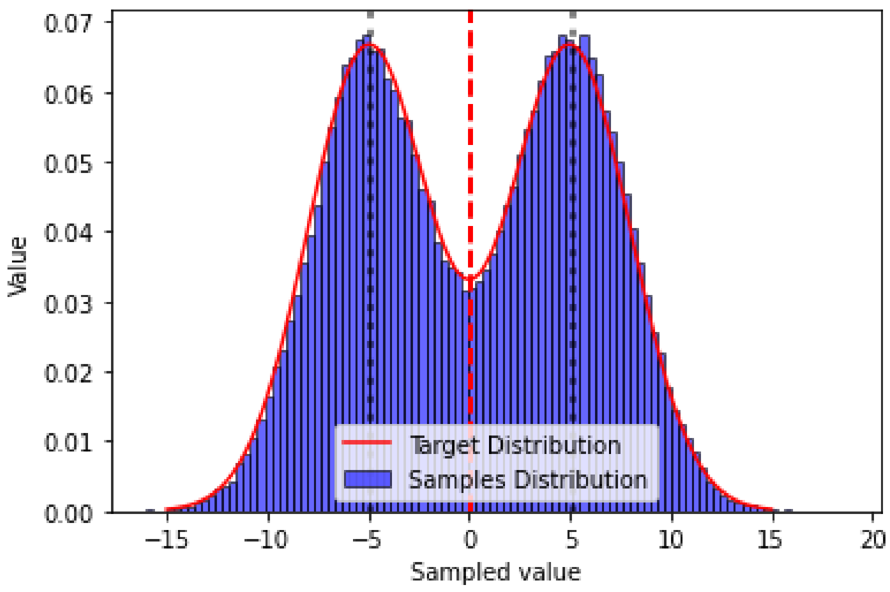

Figure 1.4: Histogram of example using Metropolis-Hasting MCMC sampling

1.2.3 After Obtaining MCMC Samples

After obtaining MCMC samples based on the algorithm introduced above, post-processing should be carried out and they are interpreted in this section. Two common post-processing of MCMC are burn-in and thinning. Burn-in, in brief, is to ignore the first b generated points. Thus, different from the customary estimation of μ in equation (1.2), for the samples $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_n$, the expectation is estimated as

$$\frac{1}{n-b}\sum_{i=b+1}^{n} f(\boldsymbol{\theta}_i)$$

The main reason to carry out burn-in is because the first few observations depend on the choice of the initial value; accordingly, they might be quite unrepresentative of the stationary distribution. The quantity of burn-in to choose is a subject issue. In this paper, the size of burn-in is determined by when the chain appears converged. Thinning to every k'th observation with k larger than 2, the estimated expectation using thinning could be represented by

$$\frac{1}{n_k}\sum_{i=1}^{n_k} f(\boldsymbol{x}_{ki})$$

The purpose of thinning responds to the concern that the samples spaced apart by k steps are almost close to being independent, whereby the dependence issue is taken into account in terms of standard errors for the sample averages. However, in spite of lower autocorrelations, variance of the estimated expectation is commonly increased due to less samples available for estimating. As the result, thinning shall not be adopted in the paper.

Another important issue regarding diagnostics within MCMC is whether the samples have neared convergence to the target distribution. Theoretically, convergence could only be achieved due to an infinite sample. Therefore, only lack of convergence could be detected, and instances which appear close to convergence are regarded as 'converged' for practical purposes. Using trace plots, if the samples seem to fluctuate around a stable value, it could be said that there is no evidence indicating a lack of convergence and this may thus be treated as having converged. More detailed convergence diagnostics could refer to [7] with a correction by [8], using summary statistics to support any judgement. The underlying concept of it is to run several independently generated Markov chains with widely different initial values. If they ends up intermingling (as can be ascertained by some statistics, then the convergence to the stationary distribution could be affirmed.

Figure 1.5 presents the trace plot for the samples plotted in Figure 1.4,whereby the red dashed line and dotted line denote the same meaning. The additional remarks to this particular case is that 200 thousands points are sampled, while 50 thousand are discard exhibiting burn-in. From the trace plot, it could be observed that samples look like a random scatter about a stable value, which supports the inference that the chain has 'converged'. As a result, these information lines are able to locate at the corresponding position.

1.2.4 More Specific MCMC Algorithms

Based on Metropolis-Hastings algorithm, the general framework for MCMC, Gibbs Sampling [9] and Hamiltonian Monte Carlo (HMC) [10] as specific instances, has been proposed; the important roles they place in reality and in fact have been well

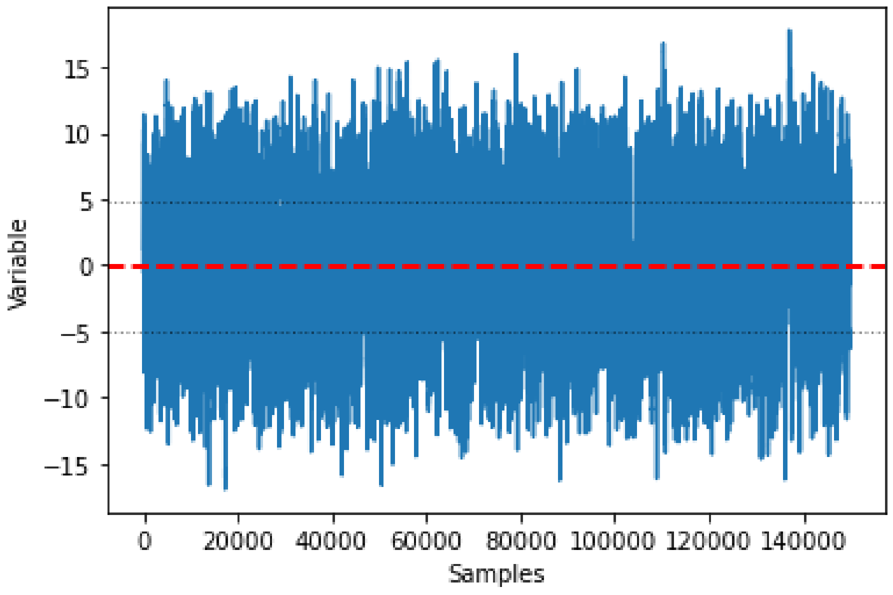

Figure 1.5: Trace plot of example using Metropolis-Hasting MCMC sampling

attested. In order to obtain a sequence of observations from multivariate probability distribution, Gibbs Sampling has potential to generate samples of one variable conditioned on other variables. Since this could be combined with analysis involving forming full-conditional distributions using factorisation theorem, it is commonly employed as a means in Bayesian inference to fit Bayesian models. The software named winBUGS along with updated version openBUGS has potential to conduct the Gibbs sampling, where BUGS represents ”Bayesian inference Using Gibbs Sampling”. The variations of Gibbs sampling include Blocked Gibbs sampling and Collapsed Gibbs sampling, with their purposes are to reduce the autocorrelation in samples whilst incurring acceptable computational costs.

Regarding Hamiltonian dynamics as the proposal function for a Markov chain, one way that physicists describe the conversion process between kinetic energy and potential energy during the motion of the objects in terms of their location and momentum, HMC could use gradient information from the target density defined by using the potential energy function to improve the exploration of the state space. Taking a series of steps depending on this first-order gradient information, this approach could potentially mitigate sensitivity to correlated parameters as well as the random walk behaviour, which plagues lots of MCMC approaches. Therefore, HMC could converge much faster than simpler methods (such as random walk Metropolis or Gibbs sampling) in sampling high-dimensional target distributions. More comprehensive introduction to the geometric tools used in HMC could refer to [11]. The variations of HMC include Riemannian Manifold Hamiltonian Monte Carlo (RMHMC) [12] which could utilize higher-order information, and shadow Hamiltonian Monte Carlo (SHMC) [13], where the acceptance rate could be improved.

Chapter 2

Background on CVs

2.1 Variance reduction

The reason why variance reduction techniques can exist is that they offer greater efficiency than obtaining more samples; accordingly, measurement of efficiency is interpreted from the beginning where this idea is proposed in [14].

2.1.1 Measuring Efficiency

For the variance reduction techniques which may be applied for samples after implementing MC methods, considering the unbiased condition and convergence rate to be $O(n^{-1/2})$ for simplicity, the error variance for sampling n observations and the cost could be written as $\frac{\sigma_1^2}{n}$ and nc_1. To meet the expected error variance τ^2, n should be set as $\frac{\sigma_1^2}{\tau^2}$ and it would cost $\frac{c_1 \cdot \sigma_1^2}{\tau^2}$. Similarly, with the error variance and cost to be $\frac{\sigma_0^2}{n}$ and nc_0, the cost of the baseline method (e.g. basic MC) should be $\frac{c_0 \cdot \sigma_0^2}{\tau^2}$. Resultantly, the efficiency of the new method against the baseline method is:

$$E = \frac{c_0 \sigma_0^2}{c_1 \sigma_1^2} \tag{2.1}$$

The efficiency could be considered to comprise of $\frac{\sigma_0^2}{\sigma_1^2}$ and $\frac{c_0}{c_1}$ where the former could be analysed theoretically, while the cost remains difficult to ascertain. For instance, it could be the computational cost, such as time measured by the programming software, which is subject to general choice. Although some details may need to be changed, this is essentially the main framework by which to measure the efficiency.

2.1.2 Criteria for Variance Reduction Techniques

General approaches to reduce the error variance include the importance of sampling (generating samples from a distinct distribution than the distribution of interest) and its extensions [15], stratified sampling (split up the domain to take samples and then combine the results) and related techniques such as post-stratification [16], antithetic variables [14] (somehow gives the opposite of sampled value to get error cancellation) and more generally (randomised) quasi-Monte-Carlo (choosing sequence of variables differently), control variates, Rao–Blackwellization [17] (characterizing the transformation of estimator to another one optimized by some criteria), Riemann sums [18] (approximating of the integral by a finite sum), and a plethora of sophisticated Markov Chain Monte Carlo (MCMC) sampling schemes [19]. Regardless of the higher levels of efficiency, all of these methods still manifest several criteria which are expected to be satisfied:

1. Unbiased: ”Unbiased” is defined as the expected value of estimator equal to the true value and under the context of posterior mean, it suggests

$$E_{p(\boldsymbol{\theta})}\left[\frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{\theta}_i)\right] = \mu$$

   in equation (1.2). It is satisfied by MC methods (not including MCMC) based on IID sampling, whereas MCMC fails about it.

2. Non-Normalised: Sampling is able to be implemented on an non-normalised density, which is main advantage of MCMC compared with MC methods (not including MCMC).

3. Fast-convergence: The convergence rate is faster than $\sqrt{n}$, which could be realised by, for example, Riemann sums mentioned above.

4. Post-process: The ability to post-process represents that the variance reduction process could be applied retrospectively, after samples have been generated, which can provide greater levels of flexibility.

The variance reducing technique applied in this thesis is characterised by control variates (CVs) where CVs based on MC and MCMC methods have the potential to satisfy the criteria 'post-process', but meet the criteria 'unbiased' and 'non-normalised', respectively.

2.1.3 Control Variates

In brief, the generic approach of control variates (CVs) for MC methods (including MCMC) is to identify a function g, corresponding to the function f as elicited in equation (1.2), such that the error variance of the estimator with f replaced by $f+g$ (equivalent to $f-g$) could be smaller, and that this g is referred to as a control variate satisfying the expectation to be zero. The detailed formula is performed here with the symbols following the previous section (notation refers to the part in (1.2)): The new estimator $\hat{\mu}_{CV}$ generated via control variates instead of $\hat{\mu}_n$ is

$$\hat{\mu}_{CV} = \frac{1}{n}\sum_{i=1}^{n}\left(f\left(\boldsymbol{\theta}_i\right)+g\left(\boldsymbol{\theta}_i\right)\right) \tag{2.2}$$

where it is satisfied that $E_{p(\boldsymbol{\theta})}[g(\boldsymbol{\theta})] = 0$ and $g(\boldsymbol{\theta})$ could be constructed by some other function $h(\boldsymbol{\theta})$, whose expectation does not have to be zero:

$$g(\boldsymbol{\theta}) = h(\boldsymbol{\theta}) - E_{p(\boldsymbol{\theta})}[h(\boldsymbol{\theta})]$$

Then, we could obtain

$$E[\hat{\mu}_{CV}] = E[\hat{\mu}_n] = \mu$$

which guarantees the unbiased property for MC methods (not including MCMC) whereby the reduced variance would be

$$\mathrm{Var}[\hat{\mu}_{CV}] = \frac{1}{n}\mathrm{Var}(f(\boldsymbol{\theta})+g(\boldsymbol{\theta})) < \frac{1}{n}\mathrm{Var}(f(\boldsymbol{\theta}))$$

The main challenge in applying CVs is related to their construction. For previous CVs, [20] built control variates for independent Metropolis-Hastings samplers, whereas [21] introduced CVs for general Metropolis-Hastings samplers. CVs are widely applied in statistics and machine learning across a range of contexts, such as topics simulating Markov processes, stochastics, reinforcement learning, variational inference and Bayesian computation – the latter of which is the purpose of this thesis.

2.2 Recent CVs Methods

Along with the evolvement in MCMC samplers, CVs methods have also been recently developed. Two recent CVs are introduced in this section, with Stein operator as the fundamental basis is interpreted from the start.

2.2.1 Stein Operator

Considering for $\boldsymbol{\theta} = (\theta_1, ..., \theta_{n_\theta})^{\mathrm{T}}$, let $\nabla_\theta = (\frac{\partial}{\partial \theta_1}, ..., \frac{\partial}{\partial \theta_{n_\theta}})^{\mathrm{T}}$ denotes the gradient and thus the Laplace Operator ∇_θ^2 would be equivalent to $\nabla_\theta \cdot \nabla_\theta$, which is $\sum_{i=1}^{n_\theta} \frac{\partial^2}{\partial \theta_i^2}$. According to the notation in [22], let U, named stein class, denotes a set of functions of $u \in U$ whose domain is $\mathbb{R}^{n_\theta}$. The defined Stein operator, written as L, originates from the method in [23] where it was used for evaluating convergence in distribution. As the integrals of the functions produced by it with respect to the target distribution $\pi(\boldsymbol{\theta})$ could be zero ($\int (Lu)(\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta} = 0$) under some necessary conditions regarding differeniablity and boundary, which corresponds to the basic condition of the control variate g that $E_{\pi(\boldsymbol{\theta})}[g(\boldsymbol{\theta})] = 0$, this property thus enables the construction of several primary CVs techniques. The choice of Stein operator is multifarious, and it could be derived based on some other proposed generator methods, or even the Schrödinger Hamiltonian in [24]. Here in terms of the CVs methods, the attention of previous research primarily focused on Langevin Stein operator [25], written as L_{L} and defined as

$$(L_{\mathrm{L}}u)(\boldsymbol{\theta}) = \nabla_\theta \cdot u(\boldsymbol{\theta}) + u(\boldsymbol{\theta}) \cdot \nabla_\theta \log(\pi(\boldsymbol{\theta})) \tag{2.3}$$

The Langevin Stein operator recovers the operator for construction in the control variates termed the control functionals [26] as well as a recent proposed neural network (NN) approach [27]. After replacing the vector field u with the gradient of the scalar-valued function v, the scalar-valued Langevin Stein operator (written as L_{SL}) could be obtained and it is defined as

$$(L_{\mathrm{SL}}v)(\boldsymbol{\theta}) = \nabla_{\theta}^{2} v(\boldsymbol{\theta}) + \nabla_{\theta} v(\boldsymbol{\theta}) \cdot \nabla_{\theta} \log(\pi(\boldsymbol{\theta})) \tag{2.4}$$

This operator is used in [28] where it is analyzed based on polynomial regression models; this approach is termed ”zero variance” (ZV). In general, approaches of CVs could be divided into parametric approaches (including ZV and NN) or non-parametric approaches, including control functionals (depending on whether the function u is parametric or not). Due to issues of complexity, practical implementations of the former are more developed than the latter [29]. Here, ZV, the typical method representing the polynomial approach (together with its adapted versions) shall be illustrated below, since it is highly relevant to this article. Further ,the called control function technique shall be briefly introduced.

2.2.2 Zero Variance Method

Zero Variance (ZV) method was firstly proposed by [24], based on the concepts of statistical mechanics, and has since been proved as efficient in variance reduction of MCMC estimators. For parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ and target distribution $\pi(\boldsymbol{\theta})$ in Bayesian computation, the constructed auxiliary function $\tilde{f}(\boldsymbol{\theta})$ to replace the original function $f(\boldsymbol{\theta})$ for sampling proposed by [24] is

$$\tilde{f}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + g(\boldsymbol{\theta}) \tag{2.5}$$

$$g(\boldsymbol{\theta}) = \frac{H[\psi(\boldsymbol{\theta})]}{\sqrt{\pi(\boldsymbol{\theta})}} \tag{2.6}$$

where $g(\boldsymbol{\theta})$ is regarded as a control variate and $H[\psi(\boldsymbol{\theta})]$ denotes the Schrödinger Hamiltonian written as

$$H[\psi(\boldsymbol{\theta})] = -\nabla_{\theta}^{2}[\psi(\boldsymbol{\theta})] + \frac{\psi(\boldsymbol{\theta})}{\sqrt{\pi(\boldsymbol{\theta})}} \nabla_{\theta}^{2}[\sqrt{\pi(\boldsymbol{\theta})}] \tag{2.7}$$

The $\psi(\boldsymbol{\theta})$ here could be any integrable function, whereas for any $\psi(\boldsymbol{\theta})$, the paper [24] reveals that $E[\frac{H[\psi(\boldsymbol{\theta})]}{\sqrt{\pi(\boldsymbol{\theta})}}] = 0$ exists, which ensures this ZV method as a kind of CVs technique that can effectively satisfy the asymptotically unbiased estimating. In addition, it could be noticed that for any $\psi_1(\boldsymbol{\theta})$, $\psi_2(\boldsymbol{\theta})$ and an arbitrary constant c, it is satisfied that $E[\frac{H[\psi_1(\boldsymbol{\theta})+c\cdot\psi_2(\boldsymbol{\theta})]}{\sqrt{\pi(\boldsymbol{\theta})}}] = 0$. As the result, the linear combination of the elements in vector $(\psi_1(\boldsymbol{\theta}), \ldots \psi_n(\boldsymbol{\theta}))^{\mathrm{T}}$ could still represent the optimal choice for constructing a control variate, which furnishes us with considerable flexibility for this ZV method. Also, due to this property, the unnecessary constant of proportionality utilised in the Hamiltonian defined above is omitted.

The polynomial form of the function $\psi(\boldsymbol{\theta})$ is proposed by [28] where the possible control variates increase concurrent to the degree of the polynomial, exponentially. Practically, it is enough to dramatically reduce the variance by using first and second polynomials [30]. According to [28], given $\psi_(\boldsymbol{\theta}) = v(\boldsymbol{\theta})\sqrt{\pi(\theta)}$ where $u(\boldsymbol{\theta})$ is the polynomial function, after inserting $\psi_(\boldsymbol{\theta})$ to the defined Hamiltonian, the control variate as well as the final form of constructed auxiliary function $\tilde{f}(\boldsymbol{\theta})$ in equation (2.5) could be obtained thus

$$g(\boldsymbol{\theta}) = \nabla^2_\theta v(\boldsymbol{\theta}) + \nabla_\theta v(\boldsymbol{\theta}) \cdot \nabla_\theta \log(\pi(\boldsymbol{\theta})) \tag{2.8}$$

$$\tilde{f}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + \nabla^2_\theta v(\boldsymbol{\theta}) + \nabla_\theta v(\boldsymbol{\theta}) \cdot \nabla_\theta \log(\pi(\boldsymbol{\theta})) \tag{2.9}$$

Although construction of the control variate in [28] does not refer to Stein operator, the final form of $g(\boldsymbol{\theta})$ is equivalent to use the scalar-valued Langevin Stein operator in equation (2.4) where scalar-valued polynomial function should be mapped.

As can be noticed that the gradient of the log-target is also the prerequisite of differential-geometric MCMC including Hamiltonian Monte Carlo (HMC), Riemann manifold Hamiltonian Monte Carlo (RMHMC), Metropolis adjusted Langevin algorithm (MALA) and the manifold Metropolis adjusted Langevin algorithm (MMALA), which are described detailedly in [12], the satisfied post-process property of CVs could support the ZV technique to facilitate implementation along with the these differential-geometric MCMC methods, so that it can help achieve convergence in fewer MCMC iterations at a moderate cost of computing time for each of the iterations. This procedure is elaborated in [30].

2.2.3 Control Functionals Method

The control functionals (CFs) method was proposed in [26] where the $g(\boldsymbol{\theta})$ defined as CF here (equivalent statistic recognized as control variate in equation (2.5) due to the fundamental conceptual distinction between two approaches) is constructed using the Langevin Stein operator appeared in equation (2.3) and u in $L_{\mathrm{L}}u$ is constrained in Hilbert space of vector fields on $\mathbb{R}^n$ to realize fully non-parametric approximation. Based on kernel methods using functional analysis, using regularized estimation, $g(\boldsymbol{\theta})$ is selected as a minimum-norm element of the Hilbert space, thereby satisfying the interpolation equations $f(\boldsymbol{\theta}_i) = g(\boldsymbol{\theta}_i) + c$ for all $i = 1, \ldots, m$ where f is the original function, m is the number of obtained $\boldsymbol{\theta}$ and $c \in \mathbb{R}$. Although this method could achieve the convergence rate faster than $\sqrt{n}$, the computational cost is $O(m^3)$ which is substantial. Further work includes [29] which combines an approach to numerical integration due to Sard, with the minimal norm interpolation construction mentioned above.

Chapter 3

Methodology and Experiments

3.0.1 The Proposed General Framework

The proposed general framework to extend zero variance (ZV) method [28] is shown in the below and it could be observed that the framework is able to encompass the existing ZV method. Considering the similarity, a control variate constructed with this proposed framework is defined as generalized zero variance method (GZV) for convenience.

The construction of ZV method could be regarded as using the scalar-valued Langevin Stein operator $[(L_{\mathrm{SL}}v)(\boldsymbol{\theta}) = \nabla_{\theta}\log(\pi(\boldsymbol{\theta})) \cdot \nabla_{\theta}\mathrm{v}(\boldsymbol{\theta}) + \nabla^2_{\theta}\mathrm{v}(\boldsymbol{\theta})]$ in equation (2.4) where $v(\boldsymbol{\theta})$ in (2.8) is set to be first or second degree polynomials. As it is mentioned that the gradient of the scalar-valued function $v : \mathbb{R}^n \to \mathbb{R}$ belongs to the stein class U that consists of function $u : \mathbb{R}^n \to \mathbb{R}^n$ described in the Langevin Stein operator (2.3), the representation of ZV method could be substituted to use the Langevin Stein operator $[(L_{\mathrm{L}}u)(\boldsymbol{\theta}) = \nabla_{\theta}\log(\pi(\boldsymbol{\theta})) \cdot \mathrm{u}(\boldsymbol{\theta}) + \nabla_{\theta} \cdot \mathrm{u}(\boldsymbol{\theta})]$ directly. To ensure a clear visualisation, the process is interpreted in matrix form as follows. Considering for $\boldsymbol{\theta} = (\theta_1, ..., \theta_n)^{\mathrm{T}}$ as parameter variable, regard $v_1(\boldsymbol{\theta})$ and $v_2(\boldsymbol{\theta})$ as the first and second degree polynomial, respectively

$$v_1(\boldsymbol{\theta}) = \sum_{i=1}^{n} a_i \theta_i$$

$$v_2(\boldsymbol{\theta}) = \sum_{i=1}^{n} a_i \theta_i + \sum_{j \leq k} a_{jk} \theta_j \theta_k$$

where a_{jk} denotes the coefficient for the θ parameter. Therefore, taking the gradient of two $v(\boldsymbol{\theta})$, the corresponding $u_1(\boldsymbol{\theta})$ and $u_2(\boldsymbol{\theta})$ would be

$$u_1(\boldsymbol{\theta}) = \begin{bmatrix} a_1 \\ \vdots \\ a_i \\ \vdots \\ a_n \end{bmatrix} ; u_2(\boldsymbol{\theta}) = \begin{bmatrix} a_1 + a_{11}\theta_1 + ... + a_{1i}\theta_i + ... + a_{1n}\theta_n \\ \vdots \\ a_i + a_{1i}\theta_1 + ... + a_{ii}\theta_i + ... + a_{in}\theta_n \\ \vdots \\ a_n + a_{1n}\theta_1 + ... + a_{in}\theta_i + ... + a_{nn}\theta_n \end{bmatrix}$$

Note that the constant value '2' produced by the derivative of square term in front of the term a_{ii} in $u_2(\boldsymbol{\theta})$, (in brief, "2" in $\nabla_\theta[a_{ii}\theta_{ii}^2] = 2a_{ii}$), is removed here and a_i in $u_1(\boldsymbol{\theta})$ and $u_2(\boldsymbol{\theta})$ only represents the coefficient of the same index but not the same value. For convenience, after defining $\theta_0 = 1$, the column vector $\boldsymbol{\theta}_+ = (\theta_0, ..., \theta_n)^\mathrm{T}$ (and treating a_i as a_{i0}), the functions could then be turned into matrix multiplication form

$$u_1(\boldsymbol{\theta}) = \boldsymbol{A}_1\boldsymbol{\theta}_+ = \begin{bmatrix} a_{10} & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i0} & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n0} & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_i \\ \vdots \\ \theta_n \end{bmatrix}$$

;

$$u_2(\boldsymbol{\theta}) = \boldsymbol{A}_2\boldsymbol{\theta}_+ = \begin{bmatrix} a_{10} & a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i0} & a_{1i} & \cdots & a_{ii} & \cdots & a_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n0} & a_{1n} & \cdots & a_{in} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_i \\ \vdots \\ \theta_n \end{bmatrix}$$

Ignoring the left column of two matrices, it could be observed that $\boldsymbol{A}_1$ is zero matrix which is a special case of $\boldsymbol{A}_2$ as symmetric matrix. Further, both two types of matrix could be considered belonging to a more general $\boldsymbol{A}$

$$\boldsymbol{A} = \begin{bmatrix} a_{10} & a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i0} & a_{i1} & \cdots & a_{ii} & \cdots & a_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{ni} & \cdots & a_{nn} \end{bmatrix} \tag{3.1}$$

Within the structure of this more general $\boldsymbol{A}$, it becomes more possible to analyze $u(\boldsymbol{\theta})$, so that the flexibility of the zero variance method could thereby be improved. In addition, θ_+ could be extended to let it be constituted by elements with regard to θ with higher order such as θ_1^3 or $\theta_1\theta_2^2$, while this would cause that the number of coefficients requiring optimization increases by exponential rate. Here, the number of parameters needed to be optimized when using $u_1(\boldsymbol{\theta})$ is n, which is $O(n)$, while that when using $u_2(\boldsymbol{\theta})$ or using $u(\boldsymbol{\theta})$ by $\boldsymbol{A}$ would be $(n^2+3n)/2$ and (n^2+n), respectively, which is $O(n^2)$.

After obtaining the expression of the control variate constructed by $u(\boldsymbol{\theta})$ with the general form $\boldsymbol{A}$ in (3.1) in order to prepare for estimation, the expression of that by some certain $u(\boldsymbol{\theta})$ like $u_1(\boldsymbol{\theta})$ or $u_2(\boldsymbol{\theta})$ could be directly concluded. For example, for $u_2(\boldsymbol{\theta})$, it is only needed to constrain a_{ki} to be equal to a_{ik} for $i;k$ in $(1,...,n)$ to conclude expression in terms of a_{ik}. Accordingly, only the procedures to obtain that expression with respect to the general $\boldsymbol{A}$ is shown, as follows.

After inserting the general $\boldsymbol{A}$ into the Langevin Stein operator, the built control variate $g(\boldsymbol{\theta}) = (L_{\mathrm{L}}u)(\boldsymbol{\theta})$ could be presented thus

$$\begin{aligned}
g(\boldsymbol{\theta}) &= \nabla_{\theta}\log(\pi(\boldsymbol{\theta}))\cdot \mathrm{u}(\boldsymbol{\theta}) + \nabla_{\theta}\cdot \mathrm{u}(\boldsymbol{\theta}) \\
&= \sum_{i=1}^{n} \frac{\partial \log(\pi(\boldsymbol{\theta}))}{\partial \theta_i} \cdot (a_{i0}\theta_0 + a_{i1}\theta_1 + \ldots + a_{in}\theta_n) + \sum_{i=1}^{n} a_{ii} \\
&= \sum_{i=1}^{n} \frac{\partial \log(\pi(\boldsymbol{\theta}))}{\partial \theta_i} \cdot \sum_{k=0}^{n} (a_{ik}\theta_k) + \sum_{i=1}^{n} a_{ii} \\
&= \sum_{i=1}^{n}\sum_{k=0}^{n} \frac{\partial \log(\pi(\boldsymbol{\theta}))}{\partial \theta_i} \cdot (a_{ik}\theta_k) + \sum_{i=1}^{n} a_{ii} \\
&= \sum_{i=1}^{n} \left[\left(\frac{\partial \log(\pi(\boldsymbol{\theta}))}{\partial \theta_i}\theta_i + 1 \right) a_{ii} + \sum_{k\neq i}^{n} \left(\frac{\partial \log(\pi(\boldsymbol{\theta}))}{\partial \theta_i}\theta_k \right) a_{ik} \right] \\
&= \sum_{i=1}^{n}\sum_{k=0}^{n} \left(\frac{\partial \log(\pi(\boldsymbol{\theta}))}{\partial \theta_i}\theta_k + \delta_{ik} \right) a_{ik}
\end{aligned}$$

where $\delta_{ik} = 1$ if $i = k$ and $\delta_{ik} = 0$ otherwise. Regarding the term in the bracket as c_{ik}, the following could be obtained

$$g(\boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{k=0}^{n} c_{ik}a_{ik} \tag{3.2}$$

$$c_{ik} = \left(\frac{\partial \log(\pi(\boldsymbol{\theta}))}{\partial \theta_i}\theta_k + \delta_{ik} \right) \tag{3.3}$$

Let $\boldsymbol{C}$ denote the matrix with the element c_{ik} in i-th row and k-th column. The matrix $\boldsymbol{C}$ and $\boldsymbol{A}$ could be stretched into vector $\boldsymbol{c}$ and $\boldsymbol{a}$ by column or row so that the control variate $g(\boldsymbol{\theta})$ would be $g(\boldsymbol{\theta}) = \boldsymbol{c}\cdot\boldsymbol{a}$. This would be helpful during the subsequent concrete computation. In addition, as it is mentioned that for some $u(\boldsymbol{\theta})$, such as $u_2(\boldsymbol{\theta})$, after setting a_{ki} to be equal to a_{ik}, it is only necessary to add up c_{ik} and c_{ki} presented by equation (3.3) to constitute the new c_{ik} corresponding to a_{ik} and thus the expression for $u(\boldsymbol{\theta})$ like $u_2(\boldsymbol{\theta})$ could be obtained. As the structure here is distinct from [28], the coefficient estimators by [28] could not be referred. Based on the expression above, the original optimization procedures for coefficients in GZV could be shown.

3.0.2 Optimize the Coefficients

Using the CVs method, the original function $f(\boldsymbol{\theta})$ is replaced by $f(\boldsymbol{\theta})+g(\boldsymbol{\theta})$. To improve the precision of estimators after MCMC in the Bayesian computation context, $g(\boldsymbol{\theta})$ could be selected by minimizing the asymptotic variance according to [31]. Here, we can obtain the optimal function $g^*(\boldsymbol{\theta}) = (L_{\mathrm{L}}u^*)(\boldsymbol{\theta})$ by solving the Stein's equation $(L_{\mathrm{L}}u^*) = f(\boldsymbol{\theta}) - E_\pi[f(\boldsymbol{\theta})]$ where the approximations are proposed in [22] by revising the Stein's equation into a variational form $J(u) = \mathrm{Var}\,[f(\boldsymbol{\theta})+(L_{\mathrm{L}}u)(\boldsymbol{\theta})]$ and minimizing the empirical approximation of this function.

Depending on m samples computed by $\pi(\boldsymbol{\theta})$ where the i-th sample written as $\boldsymbol{\theta}_i$ (not θ_i as i-th parameter variable in $\boldsymbol{\theta}$), the variance type approximation $J_m^{\mathrm{V}}(u)$ as well as the least-squares type approximation $J_m^{\mathrm{LS}}(u)$ are presented and displayed in the following
The variance type

$$J_m^{\mathrm{V}}(u) = \frac{2}{m(m-1)} \sum_{i>j} \left(f(\boldsymbol{\theta}_i) + (L_{\mathrm{L}}u)(\boldsymbol{\theta}_i) - f(\boldsymbol{\theta}_j) - (L_{\mathrm{L}}u)(\boldsymbol{\theta}_j)\right)^2 \tag{3.4}$$

The least-squares type

$$J_m^{\mathrm{LS}}(u) = \frac{1}{m} \sum_{i=1}^{m} \left(f(\boldsymbol{\theta}_i) + (L_{\mathrm{L}}u)(\boldsymbol{\theta}_i) - \frac{1}{m} \sum_{j=1}^{m} f(\boldsymbol{\theta}_j) \right)^2 \tag{3.5}$$

The least-squares type approximation is selected to compute the coefficient estimators of $g(\boldsymbol{\theta})$ in (3.2). For sake of clear interpretation, J_m^{LS} could be written in vector form. Let $\boldsymbol{f}(\boldsymbol{\theta})$ denotes the column vector with $f(\boldsymbol{\theta}_i)$ as i-th element where i is in $1,\ldots,m$ and $\boldsymbol{\mu}$ denote the column vector with size m filled with the constant $\frac{1}{m}\sum_{j=1}^{m} f(\boldsymbol{\theta}_j)$. As it is mentioned, $g(\boldsymbol{\theta})$ in (3.2) could be denoted as $g(\boldsymbol{\theta}) = (L_{\mathrm{L}}u)(\boldsymbol{\theta}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{a}$ so that the column vector filled with the sample $g(\boldsymbol{\theta}_i) = \boldsymbol{c}_i^{\mathrm{T}}\boldsymbol{a}$ (i in $1,\ldots,m$) could be written as $\bar{\boldsymbol{C}}^{\mathrm{T}}\boldsymbol{a}$ where $\bar{\boldsymbol{C}}$ is the matrix merging the sample vector $\boldsymbol{c}_i^{\mathrm{T}}$. Written as $\bar{\boldsymbol{C}}$ but not $\boldsymbol{C}$ is to avoid repetition. Finally, J_m^{LS}

again could be expressed as

$$J_m^{\mathrm{LS}}(u) = \frac{1}{m}\left[\boldsymbol{f}(\boldsymbol{\theta}) + \bar{\boldsymbol{C}}^{\mathrm{T}}\boldsymbol{a} - \boldsymbol{\mu}\right]^{\mathrm{T}}\left[\boldsymbol{f}(\boldsymbol{\theta}) + \bar{\boldsymbol{C}}^{\mathrm{T}}\boldsymbol{a} - \boldsymbol{\mu}\right] \tag{3.6}$$

Similar to solving least squares in regression problems, the minimum of $J_m^{\mathrm{LS}}(u)$ has the closed form by solving $\nabla_a J_m^{\mathrm{LS}}(u)(\boldsymbol{a}^*) = 0$, with $\boldsymbol{a}^*$ being the vector of optimal coefficient estimators. The representation of this $\boldsymbol{a}^*$ could be calculated by

$$\begin{aligned} \nabla_a J_m^{\mathrm{LS}}(u)(\boldsymbol{a}^*) &= 0 \\ \Longleftrightarrow\ 2\bar{\boldsymbol{C}}^{\mathrm{T}}\left[\boldsymbol{f}(\boldsymbol{\theta}) + \bar{\boldsymbol{C}}^{\mathrm{T}}\boldsymbol{a}^* - \boldsymbol{\mu}\right] &= 0 \\ \Longleftrightarrow\ \bar{\boldsymbol{C}}^{\mathrm{T}}\bar{\boldsymbol{C}}\boldsymbol{a}^* &= -\bar{\boldsymbol{C}}^{\mathrm{T}}\left[\boldsymbol{f}(\boldsymbol{\theta}) - \boldsymbol{\mu}\right] \\ \Longleftrightarrow\ \boldsymbol{a}^* &= -\left[\bar{\boldsymbol{C}}^{\mathrm{T}}\bar{\boldsymbol{C}}\right]^{-1}\bar{\boldsymbol{C}}^{\mathrm{T}}\left[\boldsymbol{f}(\boldsymbol{\theta}) - \boldsymbol{\mu}\right] \end{aligned} \tag{3.7}$$

In addition, the above $\boldsymbol{f}(\boldsymbol{\theta})$ and $\boldsymbol{\mu}$ is for single function f. Whereas, multiple $\boldsymbol{f}(\boldsymbol{\theta})$ and $\boldsymbol{\mu}$ sampled by different function f could be combined together and presented in matrix form. In replacing the vector form $\boldsymbol{f}(\boldsymbol{\theta})$ and $\boldsymbol{\mu}$ by this matrix form, it would be convenient to get the optimized coefficient sets for different functions of f directly. This procedure is implemented in practice.

As the representation of $J_m^{\mathrm{LS}}(u)$ is a convex function, stochastic gradient descent (SGD) could be applied when the sample size is extremely large, which would lead to the considerable computation cost of (3.7). To minimize $J_m^{\mathrm{LS}}(u)(\boldsymbol{a})$, one could iterate through $\boldsymbol{a}^{(t+1)} = \boldsymbol{a}^{(t)} - \alpha \times \nabla_a J_m^{\mathrm{LS}}(u)(\boldsymbol{a}^{(t)})$ with a suitable learning rate α. For more precise result, mini-batch and methods with respect to learning rate decay could be employed. By this step, introduction of the procedure to realize GZV method is completed.

3.0.3 Example of Failed Case

Before introducing and testing control variate constructed by GZV method above, it is necessary to mention that this approach may fail under some certain conditions. Since the control variate by $u(\boldsymbol{\theta})$ using the general A in (3.1) is novel, this case is focused and it is found to fail for dealing with samples from multivariate normal distribution. This is instead of complicated theoretical proof for arbitrary dimension, that for dimension 2 would be shown in the following. In addition, this proof could also help facilitate a comprehensive look-through and understand the implementation under the proposed framework.

The multivariate normal distribution of a 2-dimensional random vector $\boldsymbol{\theta} = (\theta_1, \theta_2)$ can be denoted by the following

$$\boldsymbol{\theta} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.8}$$

where $\boldsymbol{\mu}$ is mean and $\boldsymbol{\Sigma}$ is the covariance matrix. The pdf of this distribution could be written as

$$p(\boldsymbol{\theta}) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \tag{3.9}$$

Thus, the gradient of it is

$$\nabla_\theta [\log(\mathrm{p}(\boldsymbol{\theta}))] = -\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu}) \tag{3.10}$$

Furthermore, it would be used in construction of the control variate in formula (3.2). For convenience, the latter interpretation, $\boldsymbol{\Sigma}^{-1}$, is presented by

$$\boldsymbol{\Sigma}^{-1} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

and $\boldsymbol{\mu}$ is presented by (μ_1, μ_2). Note that $b_{12} = b_{21}$ due to the property of symmetric matrix and the simple proof for this is $\boldsymbol{\Sigma}^{-1} = (\boldsymbol{\Sigma}^{\mathrm{T}})^{-1} = (\boldsymbol{\Sigma}^{-1})^{\mathrm{T}}$. Thus equation(3.10) in matrix form would be

$$\nabla_\theta [\log(\mathrm{p}(\boldsymbol{\theta}))] = -\begin{bmatrix} b_{11}(\theta_1-\mu_1)+b_{12}(\theta_2-\mu_2) \\ b_{21}(\theta_1-\mu_1)+b_{22}(\theta_2-\mu_2) \end{bmatrix} \tag{3.11}$$

According to the formula in (3.3) and inserting the value in (3.11), the function of the stretched vector $\boldsymbol{c}_{fail}$ (by column) used for estimating the coefficients would be

$$\boldsymbol{c}_{fail} = \begin{bmatrix} c_{10} \\ c_{20} \\ c_{11} \\ c_{21} \\ c_{12} \\ c_{22} \end{bmatrix} = - \begin{bmatrix} b_{11}(\theta_1 - \mu_1) + b_{12}(\theta_2 - \mu_2) \\ b_{21}(\theta_1 - \mu_1) + b_{22}(\theta_2 - \mu_2) \\ (b_{11}(\theta_1 - \mu_1) + b_{12}(\theta_2 - \mu_2))\theta_1 + 1 \\ (b_{21}(\theta_1 - \mu_1) + b_{22}(\theta_2 - \mu_2))\theta_1 \\ (b_{11}(\theta_1 - \mu_1) + b_{12}(\theta_2 - \mu_2))\theta_2 \\ (b_{21}(\theta_1 - \mu_1) + b_{22}(\theta_2 - \mu_2))\theta_2 + 1 \end{bmatrix} \tag{3.12}$$

The fundamental reason leads up to the fault is that one of the element in the vector function $\boldsymbol{c}_{fail}$ could be linearly expressed by the other five. As the result, one column/row vector of the matrix $\bar{\boldsymbol{C}}$ in (3.7) consisting of the generated samples by $\boldsymbol{c}_{fail}$ could be linearly expressed by the other five as well,which suggests the square matrix $\left(\bar{\boldsymbol{C}}^{\mathrm{T}}\bar{\boldsymbol{C}}\right)$ is not full rank. Thus, it has no inverse and the optimization process in (3.7) is impeded, unable to be accomplished.

To verify the above statement, it is only needed to show that one element in the vector function $\boldsymbol{c}_{fail}$ could be linearly expressed by the other five. The proof is omitted while the process of it could still be briefly introduced, insofar that its concept is suitable for higher dimension multivariate normal distribution. All the elements in the right hand side of $\boldsymbol{c}_{fail}$ in (3.12) could eventually be expressed by six terms which are $(\theta_1^2, \theta_2^2, \theta_1\theta_2, \theta_1, \theta_2, 1)$. Then, $\boldsymbol{c}_{fail}$ could be written in matrix form in terms of these six terms. Depending on Gaussian elimination, it could find that one vector of that matrix form is full zero using the condition $b_{12} = b_{21}$. Resultantly, the proof with respect to one element expressed by the other five is completed. Despite the proposed framework may fail for samples when applied to samples by multivariate normal distribution, it could still be applied into most posterior distributions of Bayesian inference.

3.0.4 Design of Experiments

The proposed framework of GZV has been sufficiently discussed and thus any control variate constructed thereof should be experimented in order to assess the efficacy of the GZV method. In this section, control variate constructed by several types of $u(\boldsymbol{\theta})$ under this framework are introduced within the design of experiments. In order to visualize these vector-valued functions $u(\boldsymbol{\theta})$ more intuitively, they are presented within the general form A shown in (3.1) and called as different types of $u(\boldsymbol{\theta})$. Each polynomial element function of these types is focused on degree equal or smaller than one, which is equivalent to the second degree scalar-valued function $v(\boldsymbol{\theta})$ mapped by the scalar-valued Langevin Stein operator $[(L_{\mathrm{SL}}v)(\boldsymbol{\theta}) = \nabla_{\theta}\log(\pi(\boldsymbol{\theta}))\cdot\nabla_{\theta}\mathrm{v}(\boldsymbol{\theta}) + \nabla_{\theta}^{2}\mathrm{v}(\boldsymbol{\theta})]$ applied in the previous zero variance method.

The reason of this concentration is due to considerations for two aspects. First, it is mentioned that 1*st* or 2*nd* degree polynomial scalar-valued function by $(L_{\mathrm{SL}}v)(\boldsymbol{\theta})$ (equivalent to constant or 1*st* degree polynomial function as element in $v(\boldsymbol{\theta})$ by $(L_{\mathrm{SL}}v)(\boldsymbol{\theta})$) is sufficient in practical [28]. Second, the possible types for higher degree or even n degree are considerable and thus it would be overcomplicated for analysis and summary. Instead, choosing the degree to be smaller could offer inspiration for the degree to be higher, referring to the concept of recursive method. This is because any quantity of coefficients represented by the general form A could be equal, so as to test all the possible types is computational exhausted. Therefore, in total, eight specific types of $u(\boldsymbol{\theta})$ are experimented and they are listed in the following, where the specific feature is also briefly stated. In addition, the term 'type' is used considering that the following proposed control variates are the subdivision of all the possible control variates that could be constructed by the framework.

Type 1 of $u(\boldsymbol{\theta})$: Existed linear polynomial $v(\boldsymbol{\theta})$ by $(L_{\mathrm{SL}}v)(\boldsymbol{\theta})$ in [28]

$$u_{type1}(\boldsymbol{\theta}) = \boldsymbol{A}_{type1}\boldsymbol{\theta}_{+} = \begin{bmatrix} a_{10} & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i0} & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n0} & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_i \\ \vdots \\ \theta_n \end{bmatrix} \tag{3.13}$$

Type 2 of $u(\boldsymbol{\theta})$: Apart form left constant column, A is diagonal type

$$u_{type2}(\boldsymbol{\theta}) = \boldsymbol{A}_{type2}\boldsymbol{\theta}_{+} = \begin{bmatrix} a_{10} & a_{11} & 0 & \cdots & \cdots & 0 \\ \vdots & 0 & \ddots & & & \vdots \\ \vdots & \vdots & & \ddots & & \vdots \\ \vdots & \vdots & & & \ddots & 0 \\ a_{n0} & 0 & \cdots & \cdots & 0 & a_{nn} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \vdots \\ \vdots \\ \theta_n \end{bmatrix} \tag{3.14}$$

Type 3 of $u(\boldsymbol{\theta})$: Apart form left constant column, A is tridiagonal type (two diagonals next to the main diagonal are the same)

$$u_{type3}(\boldsymbol{\theta}) = \boldsymbol{A}_{type3}\boldsymbol{\theta}_{+} = \begin{bmatrix} a_{10} & a_{11} & a_{12} & 0 & \cdots & 0 \\ a_{20} & a_{12} & \ddots & \ddots & & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & & \ddots & \ddots & a_{n-1n} \\ a_{n0} & 0 & \cdots & 0 & a_{n-1n} & a_{nn} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \vdots \\ \theta_n \end{bmatrix} \tag{3.15}$$

Type 4 of $u(\boldsymbol{\theta})$: Apart form left constant column, A is tridiagonal type (two diagonals next to the main diagonal are different)

$$u_{type4}(\boldsymbol{\theta}) = \boldsymbol{A}_{type4}\boldsymbol{\theta}_{+} = \begin{bmatrix} a_{10} & a_{11} & a_{12} & 0 & \cdots & 0 \\ a_{20} & a_{21} & \ddots & \ddots & & \vdots \\ a_{30} & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & & \ddots & \ddots & a_{n-1n} \\ a_{n0} & 0 & \cdots & 0 & a_{nn-1} & a_{nn} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \vdots \\ \theta_n \end{bmatrix} \tag{3.16}$$

Type 5 of $u(\boldsymbol{\theta})$: All rows of A are identical

$$u_{type5}(\boldsymbol{\theta}) = \boldsymbol{A}_{type5}\boldsymbol{\theta}_{+} = \begin{bmatrix} a_{10} & a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{10} & a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{10} & a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_i \\ \vdots \\ \theta_n \end{bmatrix} \tag{3.17}$$

Type 6 of $u(\boldsymbol{\theta})$: Type 7 setting the left constant column to be zero

$$u_{type6}(\boldsymbol{\theta}) = \boldsymbol{A}_{type6}\boldsymbol{\theta}_{+} = \begin{bmatrix} 0 & a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & a_{1i} & \cdots & a_{ii} & \cdots & a_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & a_{1n} & \cdots & a_{in} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_i \\ \vdots \\ \theta_n \end{bmatrix} \tag{3.18}$$

Type 7 of $u(\boldsymbol{\theta})$: Existed second degree polynomial $v(\boldsymbol{\theta})$ by $(L_{\mathrm{SL}}v)(\boldsymbol{\theta})$ in [28]

$$u_{type7}(\boldsymbol{\theta}) = \boldsymbol{A}_{type7}\boldsymbol{\theta}_{+} = \begin{bmatrix} a_{10} & a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i0} & a_{1i} & \cdots & a_{ii} & \cdots & a_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n0} & a_{1n} & \cdots & a_{in} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_i \\ \vdots \\ \theta_n \end{bmatrix} \tag{3.19}$$

Type 8 of $u(\boldsymbol{\theta})$: Apart form left constant column, A is set to be the general form in (3.1) (non-symmetric)

$$u_{type8}(\boldsymbol{\theta}) = \boldsymbol{A}_{type8}\boldsymbol{\theta}_{+} = \begin{bmatrix} a_{10} & a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i0} & a_{i1} & \cdots & a_{ii} & \cdots & a_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{ni} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_i \\ \vdots \\ \theta_n \end{bmatrix} \tag{3.20}$$

In summary, type 1 and type 7 are existed and proposed by [28]. Type 2, type 3 and type 6 belong to the subset of the demonstrated type 7, with fewer coefficient terms. Type 4, type 5 and type 8 are constructed based on the proposed framework in the paper. In addition, all the types (except type 5 or type 6) contain the coefficient terms of type 1, while the coefficient terms of all the other types are included in type 8 which is the most general among them. After completing the illustration of different types of control variates, case studies for Bayesian computation should be prepared.

3.0.5 Models for Case Studies

Ahead of applying the proposed different types of $u(\boldsymbol{\theta})$, the form of Bayesian models and the details about these case studies first need to be interpreted in this section. Since this paper aims at implementing control variates in terms of the Bayesian field, the target distribution which needs to be sampled commonly exhibit a posterior distribution in a Bayesian model. In the Stein operator, the log-target needs to be computed and thus the log of posterior distribution should be expressed. Recall in equation (1.1), where it states that the posterior distribution of parameter is proportional to the multiplication of likelihood and prior distribution. Written in the equation form, it could be obtained thus

$$p(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{1}{c} l(\boldsymbol{x}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) \tag{3.21}$$

where $\boldsymbol{x}$ denotes variables, $\boldsymbol{\theta}$ is the corresponding parameter in the model and c is the integration of the non-normalized posterior distribution. Divergent from the representation in equation (1.1), likelihood is denoted conventionally using l while the prior uses π which is for parameters in the context. Then, the gradient of the log-posterior would be

$$\nabla_\theta \left[\log(p(\boldsymbol{\theta}|\boldsymbol{x}))\right] = \nabla_\theta \left[L(\boldsymbol{x}|\boldsymbol{\theta})\right] + \nabla_\theta \left[\log(\pi(\boldsymbol{\theta}))\right] \tag{3.22}$$

where L is the log-likelihood by convention. It could be acknowledged at this point that the unknown constant c is eliminated during taking the log, which contributes towards applying GZV methods into Bayesian computation.

Three (rather than a singular) different Bayesian models are selected to test the behaviors of these types of $u(\boldsymbol{\theta})$ in order to observe whether the behaviors relate to the concrete model. In addition, the second and the third model are suitable for the same type of variables , which allow them to be applied on the same data set. This could both bring convenience and greater control of the random effect of specific data set, an incidence that should be avoided. These models are illustrated in the following

The first model is about applying Bayesian approach to derive the unknown mean and unknown variance of the normal distribution. As the univariate normal distribution only contains 2 unknown parameters, which is insufficient for testing multiple types, here the bivariate (multivariate) normal distribution is determined, and it involves 5 unknown parameters including two known mean value and three unknown covariance value. The common non-informative joint prior distribution for the unknown mean and unknown deviation in the univariate normal distribution case is to choose the product of a uniform prior for mean and the Jeffrey's prior for the deviation. However, this is complicated to attain in a multivariate condition. Due to the fact that the purpose is for Bayesian computation but not Bayesian inference, for simplicity, the uniform prior along some range is selected for these unknown parameter. As this "model" is not meaningful to deal with real dataset but only serves as a comparison with other models, concrete deducing procedure about the gradient of log-posterior to is omitted.

The second model is Bayesian logistic regression model. In brief, still fitting the data to the logistic equation, the Bayesian logistic regression model is to estimate the coefficients through the Bayesian approach. It could quantify the uncertainty of the estimated coefficients in inferences as they are represented by the posterior distribution rather than single values by the classical logistic regression model. Examples with more detailed explanations could refer to [32]. To obtain the gradient of log-posterior, gradient of log-likelihood and log-prior are required by equation (3.22). Considering for the muti-dimensional cases, the probability of the binary response variable Y being 0 or 1 and in logistic regression is given by

$$p(y|\boldsymbol{x}_i;\boldsymbol{\theta}) = (\frac{1}{1+\exp(-\theta^{\mathrm{T}}\boldsymbol{x}_i)})^{y} \cdot (1-\frac{1}{1+\exp(-\theta^{\mathrm{T}}\boldsymbol{x}_i)})^{1-y} \tag{3.23}$$

where $\boldsymbol{x}_i$ is the i-th observed data with dimension p and $\boldsymbol{\theta}$ corresponds to the parameter. Let $\boldsymbol{X}$ denotes the $n \times p$ matrix consisting of n samples of data and $\boldsymbol{y}$ be the vector form of n samples of single y. Then the log-likelihood yields

$$\nabla_{\theta}(L(y|\boldsymbol{x}_i;\boldsymbol{\theta})) = \boldsymbol{X}^{\mathrm{T}}\left[\boldsymbol{y} - \frac{1}{1+\exp(-\boldsymbol{X}\boldsymbol{\theta})}\right] \tag{3.24}$$

Set the prior of parameter $\boldsymbol{\theta}$ to be $\boldsymbol{\theta} \sim N(\boldsymbol{0}, 100\boldsymbol{I})$ where $\boldsymbol{I}$ is the identity matrix by referring to the case in [30], which is a common non-informative matrix for the parameter whose range is in $\mathbb{R}$. Then the gradient of log-prior would be

$$\nabla_\theta(\log(\pi(\boldsymbol{\theta}))) = -\frac{1}{100}\boldsymbol{\theta} \tag{3.25}$$

According to the the deduced formula of log-posterior in equation (3.22), it would be

$$\nabla_\theta(\log(p(\boldsymbol{\theta}|\boldsymbol{y};\boldsymbol{X}))) = \boldsymbol{X}^{\mathrm{T}}\left[\boldsymbol{y} - \frac{1}{1+\exp(-\boldsymbol{X}\boldsymbol{\theta})}\right] - \frac{1}{100}\boldsymbol{\theta} \tag{3.26}$$

This derived posterior distribution could be used to obtain samples of parameters so as to conduct Bayesian inference.

The third model is Bayesian probit regression model. Similar to the logistic regression model, it is also a popular specification for the binary response model where the probit model employs a probit link function Φ^{-1} (the inverse cumulative distribution), while the logistic model uses a logit link function. Using the same notation in the description of Bayesian logistic model, the probability of y would be

$$p(y|\boldsymbol{x}_i;\boldsymbol{\theta}) = \Phi(\theta^{\mathrm{T}}\boldsymbol{x}_i)^y \cdot \Phi(-\theta^{\mathrm{T}}\boldsymbol{x}_i)^{1-y} \tag{3.27}$$

whence the log-likelihood could be derived

$$\nabla_\theta(L(y|\boldsymbol{x}_i;\boldsymbol{\theta})) = \sum_{i=1}^{n}\left[y_i(\log(\Phi(\theta^{\mathrm{T}}\boldsymbol{x}_i)) + (1-y_i)\log(\Phi(-\theta^{\mathrm{T}}\boldsymbol{x}_i))\right] \tag{3.28}$$

Remaining the same prior of the parameters as the logistic model before and hence the log-prior would be (3.25). Finally, the gradient of log-posterior could be given by

$$\nabla_\theta(\log(p(\boldsymbol{\theta}|\boldsymbol{y};\boldsymbol{X}))) = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{a} - \frac{1}{100}\boldsymbol{\theta} \tag{3.29}$$

where the $i-th$ element of the p-dimensional column vector $\boldsymbol{a}$ is

$$a_i = \frac{y_i N(\theta^{\mathrm{T}}\boldsymbol{x}_i)}{\Phi(\theta^{\mathrm{T}}\boldsymbol{x}_i)} - \frac{(1-y_i)N(-\theta^{\mathrm{T}}\boldsymbol{x}_i)}{\Phi(-\theta^{\mathrm{T}}\boldsymbol{x}_i)} \tag{3.30}$$

Note that N above presents the density of standard normal distribution. All the log-posterior of the three models for sampling are prepared during this stage.

Different from other datasets, the observed samples for the first model (bivariate normal distribution) is not real, but generated by the predetermined value of parameters. The advantage is that it would be possible to check the concluded Bayesian information (estimated mean value of parameters) and compare it with the real value. In this case study, 1000 samples are generated artificially as the dataset for the bivariate normal distribution $\boldsymbol{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ ($\boldsymbol{X}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ denotes variables, mean and covariance matrix, respectively), $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are set to be

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \end{bmatrix}; \Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix} \tag{3.31}$$

Two real cases are picked up to apply the Bayesian logistic regression model as well as the Bayesian probit regression model so that 5 case studies could be generated in total. Apart from the label variable, the dimension of independent variables of two cases are different and it could thus help compare the behaviors of types of $u(\boldsymbol{\theta})$ under different dimension of $\boldsymbol{\theta}$. The first case is from the website with regard to introduce the computational statistics in Python [33], which includes observations of height, weight and the binary label sex for prediction. Including the constant coefficient, dimension $(n \times p)$ of the design matrix $\boldsymbol{X}$ in equation (3.26) and (3.29) is (70×3), which suggests the dimension of $\boldsymbol{\theta}$ for sampling would be 3. This case is abbreviated to HtWt for short. The second case is taken from [34] and also referred to by [30] in the existed ZV method where the four covariates, as the measurements of the Swiss banknotes, are Length: Length of bill (mm), Left: Width of left edge (mm), Right: Width of right edge (mm) and Bottom: Bottom margin width (mm). Further, the binary response variable is aimed at detecting whether a banknote is counterfeit (1) or genuine (0). The size of design matrix $\boldsymbol{X}$ is (200×5) (including constant coefficient) and thus dimension of $\boldsymbol{\theta}$ would be 5. This case is abbreviated to Swiss for short.

During the concrete implementation, sampling is carried out in Python with a package for Bayesian inference named PyStan which is a Python interface to the Stan [35], a probabilistic programming language. As explained in the introduction to Stan, it is a state-of-the-art platform for statistical modeling and is able to realize high-performance statistical computation. Despite the codes being written in python, computation is run in C++, which could ensure high computational speed. The key advantage of using PyStan is that it conducts the No-U-Turn sampler [36], which is an extension to Hamiltonian Monte Carlo (HMC) sampling. Moreover, given the common benefits of HMC mentioned in section 1.2.4, the log-gradient could be returned for the continued GZV method and it is tested that this returned value yields the calculated one utilising the log-posterior formula. This process is similar to what described in [30] with regard to applying zero variance method after the several differential geometric MCMC, whereas the propose here is to investigate the multiple types of $u(\boldsymbol{\theta})$ and it being distinct fundamentally. Therefore, it is necessary to assume the GZV method behaves independently from the various MCMC samplers. In addition, not only PyStan, but another powerful Python package called PyMC3 [37] is also suitable in the production of similar samples (by HMC) and carries out the tasks in the paper, whilst not requiring to write in C++ synteax; however, this may not result in better performance. The extremely lightweight python package, called emcee, that conducts basic MCMC sampling is also for such task, but it does not have much built-in functionally, such as specially-defined objects for common existed distributions (i.e. normal) as the other two, which suggests it may not be ideal.

The generated samples from these five case studies could not be compared directly and thus criteria to assess these samples in terms of the variance reduction should be stated.

3.0.6 Criteria to Assess the Proposed types

After samples is obtained from the MCMC (HMC) sampler, in preference to testing these types of $u(\boldsymbol{\theta})$, the priority is to check the convergence and decide the size of burn-in. In the paper, instead of using summary statistics, trace plots are displayed for each case study and the size of burn-in is determined by observing when the iterations fluctuate steadily along a constant value. To avoid issue about the influence of the initial point, various independent chains are generated and all the trace plots are checked. For all the case studies, there is evidence to suggest that samples could converge; details about size of burn-in would be explained later. Figure 3.1 is an instance of the trace plot (after burn-in) in the bivariate normal case study. Furthermore, it could also be noticed that the middle value of these lines in the trace

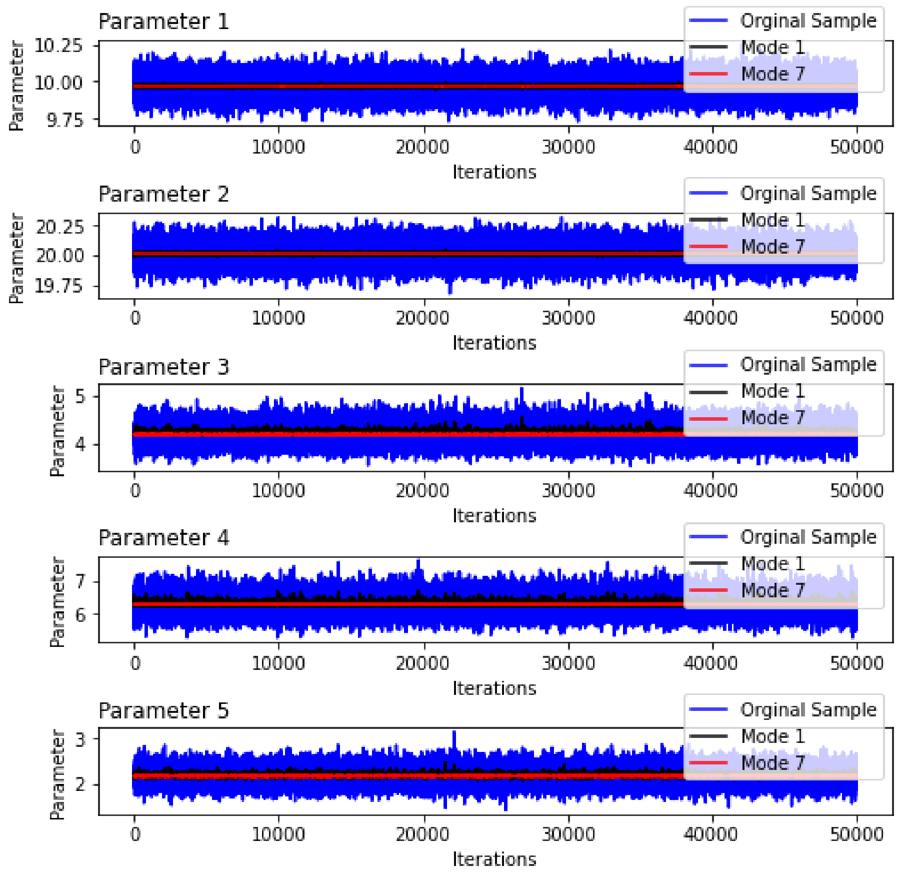

Figure 3.1: Trace plot of the original samples (blue line), applying type 1 (black line) type 7 (red line) generalized zero variance method. Parameter 1,...,5 are μ_1, μ_2, σ_{11}, σ_{22}, σ_{12}, respectively, according to equation (3.31)
. 'Modes' in the plot corresponds to 'Type'.

is close to the set-up value of these parameters, which verifies the correction of the implemented types for reducing variance.

With respect to assess types of $u(\boldsymbol{\theta})$, two parts concerned are variance reduction and the corresponding computational cost. In terms of variance reduction, considering that it is hard to comment on a certain value of estimated variance, resultantly, the ratio of estimated variance after implementing control variates to that of the original samples would be recorded instead, whereby it would be defined as 'variance reduction ratio' (VRR) in the paper. In addition, another method to quantify the variance reduction is called "variance reduction factor" in [30], yet its key concept is called 'initial monotone sequence estimator' and was firstly proposed in [38]. However, it would not be adapted for two reasons. First. it is not compatible to the 'measuring efficiency' (illustrated later), and second, due to hardware issue, there could not be enough independent chains, with as many as [30] (100 chains) to be generated, so that the advantage of this assessment method could be sufficiently displayed. As a minimum, this emphasises the necessity of multiple independent chains and for each case study, 5 independent chains are sampled, hence the average of any recorded value shall be adopted. In addition, in order to visualize the reduced variance, boxplots are also drawn for different types in all these cases studies.

In terms of computational cost, it is represented by the program running time, which is the summation of cpu running time, time for IO operations and waiting. As the number of coefficients needed to be estimates would affect the computational cost, it is also recorded for each type. The running time for the original samples only contains the generating time, while that for each type to carry out control variates method (generalized zero variance method) includes three steps: generating samples, constructing matrix $\bar{\boldsymbol{C}}$ in (3.6) (coded as tensor actually) and optimizing the coefficients in general. As the matrix $\bar{\boldsymbol{C}}$ for all the possible coefficients is constructed initially and columns are then allocated for the corresponding type, except for type 8, the time taken by other types should only depends on the concrete coefficient that they use. Therefore, the running time of each type is evaluated by $(time\ for\ constructing\ matrix) \times (number\ of\ coefficients\ in\ the\ type)/(number\ of\ coefficients\ in\ type\ 8)$.

Due to the trade-off between variance reduction and the corresponding computational cost, they could not be considered separately. Instead, the measuring efficiency (ME) illustrated previously in equation (2.1) could be applied. In brief, it compares the cost of two approaches in the form of ratio when the same variance of the expected statistic is reached. Let cost of the original sampling and type i be denoted by c_0 and c_{type-i} and estimated variance be $\hat{\sigma}_0^2$ and $\hat{\sigma}_{type-i}^2$, , respectively. Then the measuring efficiency (ME) in ratio form is

$$E_{type-i}^{0} = \frac{c_0 \cdot \hat{\sigma}_0^2}{c_{type-i} \cdot \hat{\sigma}_{type-i}^2} \tag{3.32}$$

and it could be called that type i is E_{type-i}^{0} times more efficient than the original sampling. In addition, this form offers greater convenience to compare the efficiency between any type i and type j by

$$\frac{E_{type-i}^{0}}{E_{type-j}^{0}} = \frac{\frac{c_0 \cdot \hat{\sigma}_0^2}{c_{type-i} \cdot \hat{\sigma}_{type-i}^2}}{\frac{c_0 \cdot \hat{\sigma}_0^2}{c_{type-j} \cdot \hat{\sigma}_{type-j}^2}} = \frac{c_{type-j} \cdot \hat{\sigma}_{type-j}^2}{c_{type-i} \cdot \hat{\sigma}_{type-i}^2} = E_{type-i}^{type-j} \tag{3.33}$$

With the clear justified VRR and ME as criteria, results could be shown and analyzed.

3.0.7 Results and Analysis

Results of the 5 case studies resulting from the assessment detailed are generated, while only one result of the case study (Bayesian logistic model + Swiss dataset) is displayed and the four results are put in appendix B.

Case study 3: Bayesian logistic model + Swiss dataset

Type	θ_1 VRR	θ_2 VRR	θ_3 VRR	θ_4 VRR	θ_5 VRR
type 1	4.912e-02	3.637e-02	3.551e-02	7.974e-02	2.512e-03
type 2	2.879e-02	3.166e-02	2.564e-02	2.578e-02	1.876e-03
type 3	2.406e-02	2.251e-02	2.086e-02	1.174e-02	4.776e-04
type 4	6.584e-03	7.783e-03	1.541e-03	6.251e-03	3.064e-04
type 5	9.998e-01	9.996e-01	9.994e-01	9.997e-01	9.997e-01
type 6	1.739e-01	2.373e-01	2.069e-01	1.289e-02	8.069e-01
type 7	7.713e-04	1.030e-03	1.035e-03	8.300e-04	1.463e-04
type 8	5.215e-04	4.694e-04	4.366e-04	7.624e-04	7.887e-05

Table 3.1: Table shows the variance reduction ratio (VRR) of each type in sampling the posterior distribution of each parameter.

Type	*Number of Coefficients*	*Running time (s)*
type 1	5	6.644e-03
type 2	10	9.309e-03
type 3	14	1.283e-02
type 4	18	1.536e-02
type 5	6	7.780e-03
type 6	15	1.396e-02
type 7	20	1.662e-02
type 8	30	2.493e-02

Table 3.2: Tables shows the number of coefficients that needed to be estimated and running time (apert from the original sampling) of each type in the generalized zero variance method. The original sampling spends 1494.627 seconds

Type	θ_1 ME	θ_2 ME	θ_3 ME	θ_4 ME	θ_5 ME
type 1	2.036e+01	2.750e+01	2.816e+01	1.254e+01	3.981e+02
type 2	3.474e+01	3.159e+01	3.900e+01	3.879e+01	5.331e+02
type 3	4.156e+01	4.443e+01	4.794e+01	8.516e+01	2.094e+03
type 4	1.519e+02	1.285e+02	6.489e+02	1.600e+02	3.264e+03
type 5	1.000e+00	1.000e+00	1.001e+00	1.000e+00	1.000e+00
type 6	5.752e+00	4.214e+00	4.833e+00	7.755e+01	1.239e+00
type 7	1.297e+03	9.707e+02	9.659e+02	1.205e+03	6.834e+03
type 8	1.917e+03	2.130e+03	2.291e+03	1.312e+03	1.268e+04

Table 3.3: Table shows the measuring efficiency (ME) of each type in sampling the posterior distribution of each parameter.

Prior to any analysis of the results, it needs to be mentioned that for all these case studies, 55,000 samples are generated and the first 5000 are discard for burn-in. This processing in Bayesian logistic/probit model with HtWt dataset is same to [30]. When measuring the computational cost, it is assumed that computation cost of each sample by the same variable is almost identical, so that ME as a criteria described in [14] could be applied. In this thesis, it is also assumed that computational cost of each sample by different variable shall be almost equal, not only for sake of saving time on coding and implementing the codes, but because of the scope for error if sampling each variable independently by PyStan. However, this assumption may not be quite reliable in normal case, since the parameter variables of posterior is not exchangeable. In addition, each value appeared in these tables is the average of five independent experiments under the same case study for promoting accuracy.

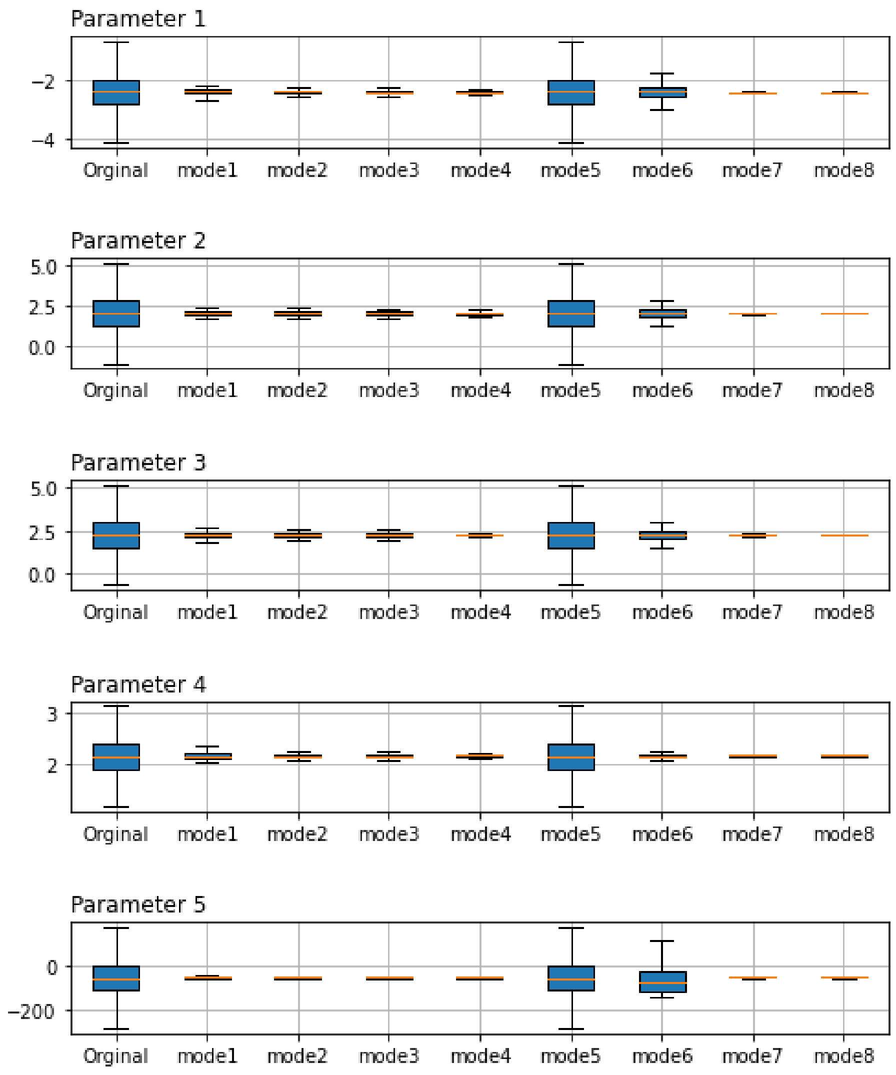

Figure 3.2: Boxplot of each parameter in the original samples as well as samples after implementing 8 types. (P.S. mode'in the plot corresponds to 'type' in the thesis)

The boxplot is the plotted only for the first experiment, for visualization, and thus should be seen together with the tables due to its inaccuracy.

After completing the complement of details about these case studies, general observations from the results along with brief analysis could be listed:

1. Computational cost between sampling and CVs method: The computation cost in conducting the CVs method is far smaller than sampling, especially when the model is complicated (comparing running time of Bayesian logistic/probit model case (Table 3.2 & B.11) with that of normal case (Table B.2) where the number of parameter variables are both 5). This means that the

ME almost becomes reciprocal of the corresponding VRR. In addition, this feature also suggests the necessity to use CVs methods.

2. General trend: It could be seen that in all these five studies, control variates constructed by different types of $u(\boldsymbol{\theta})$ exhibit similar variance reduction trends under the same parameter variables especially when comparing exchangeable parameters. In general, VRR/ME decreases/increases from type 1 to type 4 as well as from type 6 to type 8. In addition, type 5 performs the worst, having nearly no variance reduction ability in most cases. In contrast, type 8 could reduce the variance of estimator considerably than the other 7 except. Moreover, control variate types with more coefficients to be estimated relates to smaller/larger VRR/ME, insofar more coefficients correspond to greater flexibility of control variate. (P.S. tables should refer to B.6, 3.3, B.3, B.9 and B.12)

3. Special phenomenon: Different from other case studies, it could be noticed that type 1 control variate does not reduce much variance in samples of parameter 3 merely during in experiment (see table B.7 and boxplot B.3); meanwhile, type 5 behaves particularly efficient in this experiment. As there are not enough controlled trials, this paper is unable to conclude when such features originate. Moreover, type 8 reduces variance merely inefficiently in normal case (see table B.1 and boxplot B.1). More normal cases should be experimented by type 8 to check whether type 8 fails for dealing with samples of mean or deviation parameter in posterior distribution.

To analyze the behaviors of control variates constructed by different types of $u(\boldsymbol{\theta})$ where the the designs of them are illustrated in section 3.0.4, rather than interpreting them one by one, discussing several of them together according to their relations should be more meaningful. The relations should also refer to that section (3.0.4).

1. Type 7 vs. type 8: Type 7 and type 8 are the most efficient type of control variates by ZV method and GZV method, respectively. The results show

that type 8 is even more efficient that type 7, which suugests the existed ZV method is promoted by GZV method. (Refer to table B.6, 3.3, B.9, B.12)

2. From type 1 to type 4; From type 7 and type 8: Within the first four types, the latter can encompass the former by increasing the flexibility of coefficients on the diagonal line in $\boldsymbol{A}$ matrix (see (3.1)). Keeping adding coefficients on the diagonal line or revising the coefficients on some diagonals not to be symmetric (increase the flexibility) following this order, the final two types would be type 7 and type 8. Considering that type 1 and type 7 are the only two approaches of constructing control variates in [28] as a first-degree polynomial approach and second-degree polynomial approach (respectively), intermediate control variates between these two approaches could be created referring to the aforementioned process, for adding or revising coefficients on the diagonal lines. The reliability for such intermediate control variates could be supported by the fact that VRR is decreased steadily from type 1 to type 4. (Refer to table B.4, 3.1, B.7, B.10)

3. Type 1 vs. type 5: Type 5 has one more coefficents than type 1 (refer to table B.5, 3.2, B.2) but generally behaves terrible in variance reduction. This comparison aims at showing how more coefficients for constructing control variates do not necessarily facilitate higher variance reduction. (Refer to table B.4, 3.1, B.1, B.7, B.10)

4. (Type 3 vs. type4) & (type 7 vs. type8): Both type 4 & type8 are more flexible versions of type 3 & type 7 by revising the coefficients on the diagonal line not to be symmetric. Whereas, the number of coefficients needs to be estimated are $O(n)$ in type 3 & type 4 and $O(n^2)$ in type 7 & type 8 where n is the quantity of parameter variables. According to the ME tables from the case studies, it could be noticed the improvement of efficiency between type 3 & type 4 is more drastic than that between type 7 & type 8. If this feature still exists when the elements of $u(\boldsymbol{\theta})$ mentioned in section 3.0.4 consist of higher degrees of polynomials ($\iff$ $\boldsymbol{\theta}_+$ in section 3.0.4 is not limited to be first

degree), this could inspire some other types of efficient control variates to be created (Refer to table B.4, 3.1, B.7, B.10).

5. Type 1, type 6 and type 7: Type 7 could be considered as combination of type 1 and type 6 in terms of the consisted coefficients, whereas the ME of type 7 is far more than the summation of the other two. This characteristic suggests that the existence of interaction between coefficients (Refer to table B.4, 3.1, B.1, B.7, B.10).

The analysis about these eight types of control variates by GZV has been completed.

Chapter 4

General Conclusions

To estimate the quantity of interest in Bayesian inference, the problem is that generating samples using MCMC technique () may not be efficient enough such that the precision of estimator is not ideal enough. Existed paper proposed ZV method (included in CVs method), which could carry out variance reduction to increase that precision under suitable computational cost. This thesis has proposed a more general framework for encompassing and extending the existed ZV method by considering another type of Stein operator. Within this framework, multiple types of control variates, including the novel ones, are constructed in order to promote and extend the mere two in the ZV method. Experimenting on 5 case studies by employing Bayesian models, it shows that in general, the constructed types with more coefficients to be estimated are more flexible and thus they could reduce more variance of the estimators. Despite more computational cost requirements for estimating these coefficients, these costs are trivial compared with that needed for sampling and thus the most general type of (novel) control variate could even be more efficient than the types of control variate existed in ZV method. In addition, depending on comparing the related types, the approach to create intermediate control variates between the two in ZV method is stated and some features are also summarized. On the whole, this thesis contributes to the direction of ZV method and it leaves open a wide space with a reliable framework, as well as generating inspirations for further development of ZV method. Nevertheless, the thesis still contains some limitations that may need to be improved:

1. Specifically, an analysis of different types of control variates requires more theoretical investigation such that the fundamental reason why some control variates behaves efficiently could be explained.

2. It is observed that the proposed framework would fail for dealing with normal distribution. The sufficient condition for dealing with other models shall be concluded.

3. The constructed types are restricted in terms of the degrees of the vector-valued function that needs to be mapped by this Stein operator. Further comparisons between the constructed types and types with no limit are needed; any such comparison could still be carried out within the proposed framework.

4. Further, the thesis only investigates on Bayesian logistic/probit model and a informal normal model. More should be experimented with.

5. Finally, the estimator whose variance should be reduced is the expectation of each parameter variable, while more estimators could be tested.

Appendix A

Appendix About Codes

Codes could refer to the link in github: Link

Appendix B

Appendix About Plots and Tables

Case study 1: Bivariate normal model + Generated bivariate normal dataset

Type	θ_1 VRR	θ_2 VRR	θ_3 VRR	θ_4 VRR	θ_5 VRR
type 1	3.007e-03	3.062e-03	2.082e-02	2.125e-02	1.807e-02
type 2	3.006e-03	3.061e-03	1.551e-03	1.542e-03	1.435e-02
type 3	3.006e-03	3.060e-03	7.930e-04	7.864e-04	7.124e-03
type 4	3.005e-03	3.059e-03	9.383e-05	1.082e-04	6.218e-04
type 5	2.597e-01	5.059e-01	9.102e-01	9.606e-01	8.994e-01
type 6	9.897e-04	1.096e-03	1.378e-03	2.483e-03	2.428e-03
type 7	5.169e-07	2.955e-07	3.743e-05	3.914e-05	3.503e-05
type 8	1.178e+01	9.469e+00	1.361e-01	7.562e-01	1.323e-01

Table B.1: Table shows the variance reduction ratio (VRR) of each type in sampling the posterior distribution of each parameter.

Type	*Number of Coefficients*	*Running time (s)*
type 1	5	7.972e-03
type 2	10	1.096e-02
type 3	14	1.274e-02
type 4	18	1.852e-02
type 5	6	9.166e-03
type 6	15	1.395e-02
type 7	20	1.991e-02
type 8	30	2.688e-02

Table B.2: Tables shows the number of coefficients that needed to be estimated and running time (apert from the original sampling) of each type in the generalized zero variance method. The original sampling spends 438.583 seconds

Type	θ_1 ME	θ_2 ME	θ_3 ME	θ_4 ME	θ_5 ME
type 1	3.326e+02	3.266e+02	4.803e+01	4.706e+01	5.534e+01
type 2	3.327e+02	3.267e+02	6.447e+02	6.485e+02	6.969e+01
type 3	3.327e+02	3.268e+02	1.261e+03	1.272e+03	1.404e+02
type 4	3.327e+02	3.269e+02	1.066e+04	9.246e+03	1.608e+03
type 5	3.851e+00	1.977e+00	1.099e+00	1.041e+00	1.112e+00
type 6	1.010e+03	9.125e+02	7.257e+02	4.028e+02	4.118e+02
type 7	1.934e+06	3.384e+06	2.672e+04	2.555e+04	2.855e+04
type 8	8.487e-02	1.056e-01	7.349e+00	1.322e+00	7.561e+00

Table B.3: Table shows the measuring efficiency (ME) of each type in sampling the posterior distribution of each parameter.

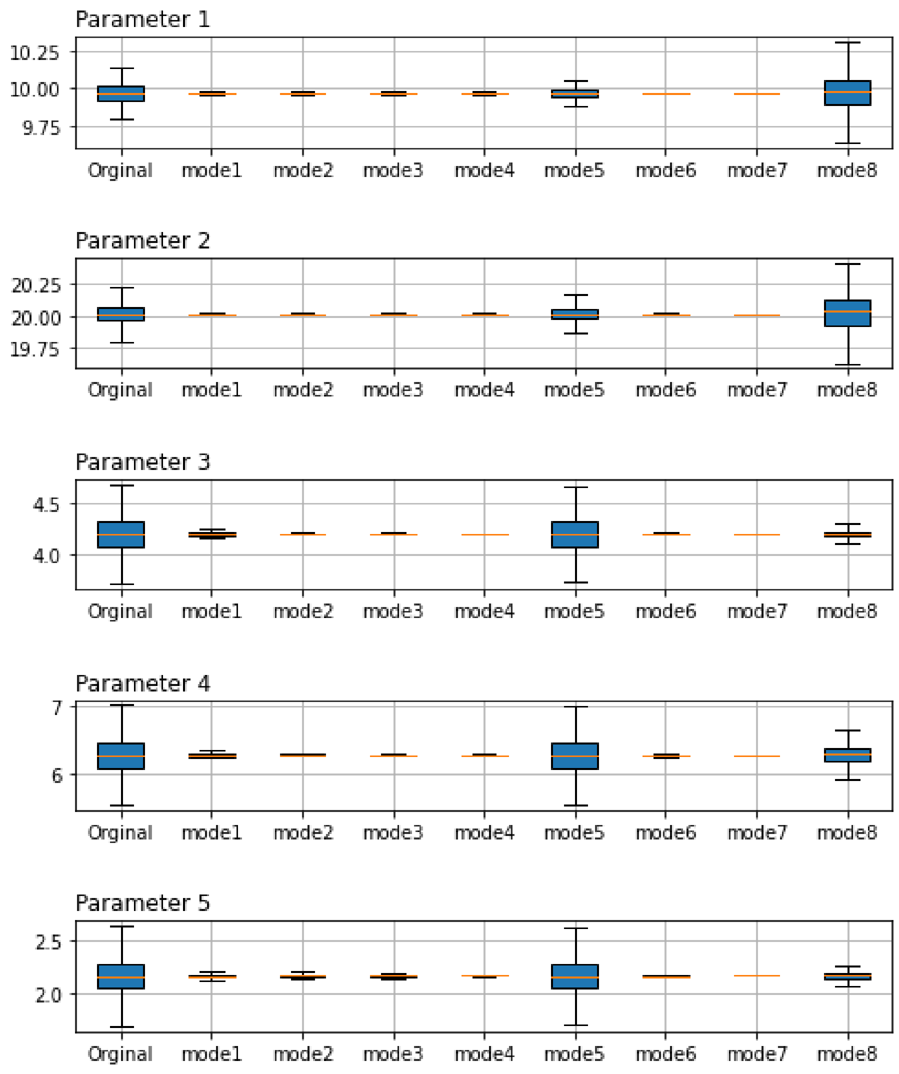

Figure B.1: Boxplot of each parameter in the original samples as well as samples after implementing 8 types. (P.S. mode'in the plot corresponds to 'type' in the thesis)

Case study 2: Bayesian logistic model + HtWt dataset

Type	θ_1 VRR	θ_2 VRR	θ_3 VRR
type 1	3.636e-02	1.092e-01	1.157e-01
type 2	3.140e-02	2.444e-02	2.441e-02
type 3	3.862e-03	1.156e-03	1.308e-03
type 4	4.226e-04	1.004e-03	9.660e-04
type 5	9.837e-01	9.996e-01	9.996e-01
type 6	2.564e-03	3.211e-03	2.593e-03
type 7	4.244e-04	1.128e-03	1.083e-03
type 8	3.867e-04	9.821e-04	9.493e-04

Table B.4: Table shows the variance reduction ratio (VRR) of each type in sampling the posterior distribution of each parameter.

Type	*Number of Coefficients*	*Running time (s)*
type 1	3	5.231e-03
type 2	6	5.484e-03
type 3	8	6.645e-03
type 4	10	7.810e-03
type 5	4	4.339e-03
type 6	6	6.461e-03
type 7	9	6.461e-03
type 8	12	8.973e-03

Table B.5: Tables shows the number of coefficients that needed to be estimated and running time (apert from the original sampling) of each type in the generalized zero variance method. The original sampling spends 84.404 seconds

Type	θ_1 ME	θ_2 ME	θ_3 ME
type 1	2.750e+01	9.156e+00	8.644e+00
type 2	3.185e+01	4.091e+01	4.096e+01
type 3	2.589e+02	8.652e+02	7.647e+02
type 4	2.366e+03	9.962e+02	1.035e+03
type 5	1.017e+00	1.000e+00	1.000e+00
type 6	3.899e+02	3.114e+02	3.856e+02
type 7	2.356e+03	8.866e+02	9.234e+02
type 8	2.585e+03	1.018e+03	1.053e+03

Table B.6: Table shows the measuring efficiency (ME) of each type in sampling the posterior distribution of each parameter.

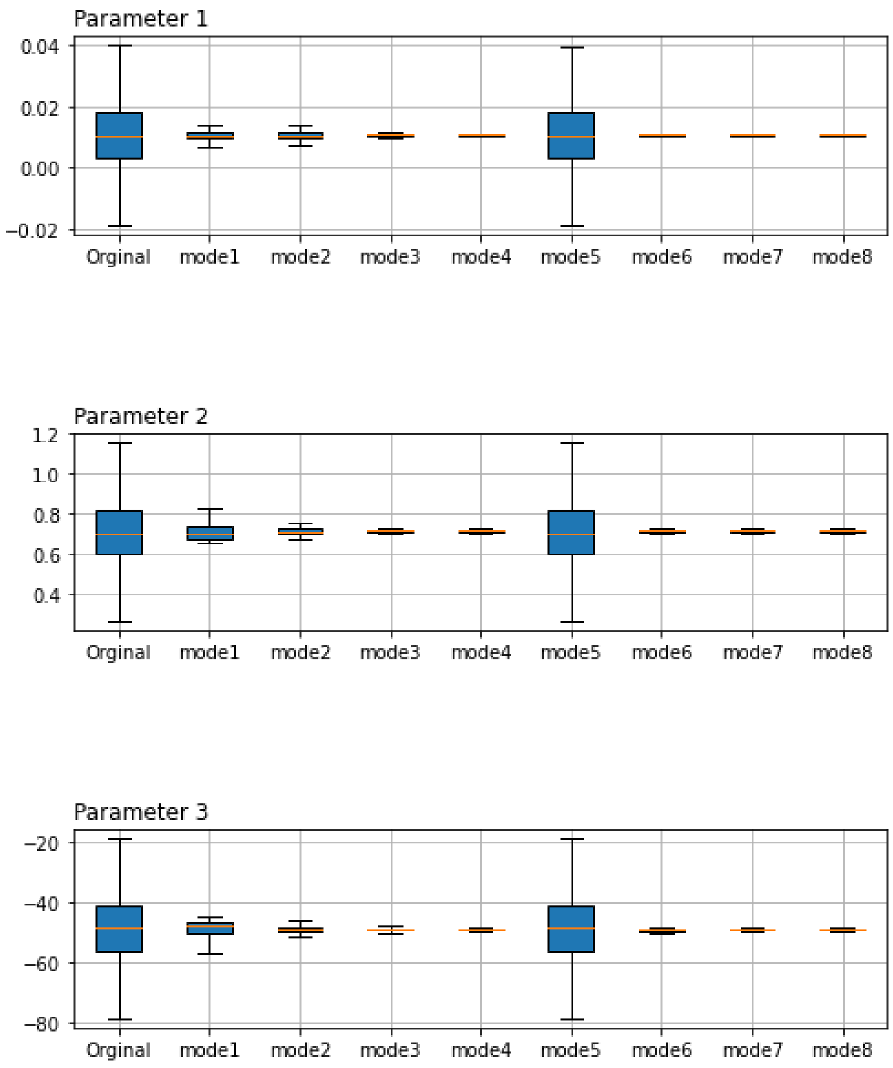

Figure B.2: Boxplot of each parameter in the original samples as well as samples after implementing 8 types. (P.S. mode'in the plot corresponds to 'type' in the thesis)

Case study 4: Bayesian probit model + HtWt dataset

Type	θ_1 VRR	θ_2 VRR	θ_3 VRR
type 1	2.738e-06	3.739e-06	8.781e-01
type 2	8.899e-07	6.650e-07	1.644e-04
type 3	4.695e-07	5.030e-07	2.079e-04
type 4	1.550e-07	1.598e-07	1.835e-04
type 5	1.665e-04	1.663e-04	1.901e-04
type 6	2.343e-06	1.910e-06	1.408e-03
type 7	1.758e-06	1.906e-06	2.458e-04
type 8	5.792e-07	6.084e-07	5.132e-05

Table B.7: Table shows the variance reduction ratio (VRR) of each type in sampling the posterior distribution of each parameter.

Type	*Number of Coefficients*	*Running time (s)*
type 1	3	4.233e-03
type 2	6	5.483e-03
type 3	8	5.650e-03
type 4	10	7.806e-03
type 5	4	4.320e-03
type 6	6	5.483e-03
type 7	9	7.724e-03
type 8	12	7.973e-03

Table B.8: Tables shows the number of coefficients that needed to be estimated and running time (apert from the original sampling) of each type in the generalized zero variance method. The original sampling spends 48.539 seconds

Type	θ_1 ME	θ_2 ME	θ_3 ME
type 1	3.653e+05	2.674e+05	1.139e+00
type 2	1.124e+06	1.503e+06	6.081e+03
type 3	2.130e+06	1.988e+06	4.810e+03
type 4	6.449e+06	6.256e+06	5.449e+03
type 5	6.007e+03	6.013e+03	5.260e+03
type 6	4.268e+05	5.235e+05	7.102e+02
type 7	5.687e+05	5.246e+05	4.067e+03
type 8	1.726e+06	1.643e+06	1.948e+04

Table B.9: Table shows the measuring efficiency (ME) of each type in sampling the posterior distribution of each parameter.

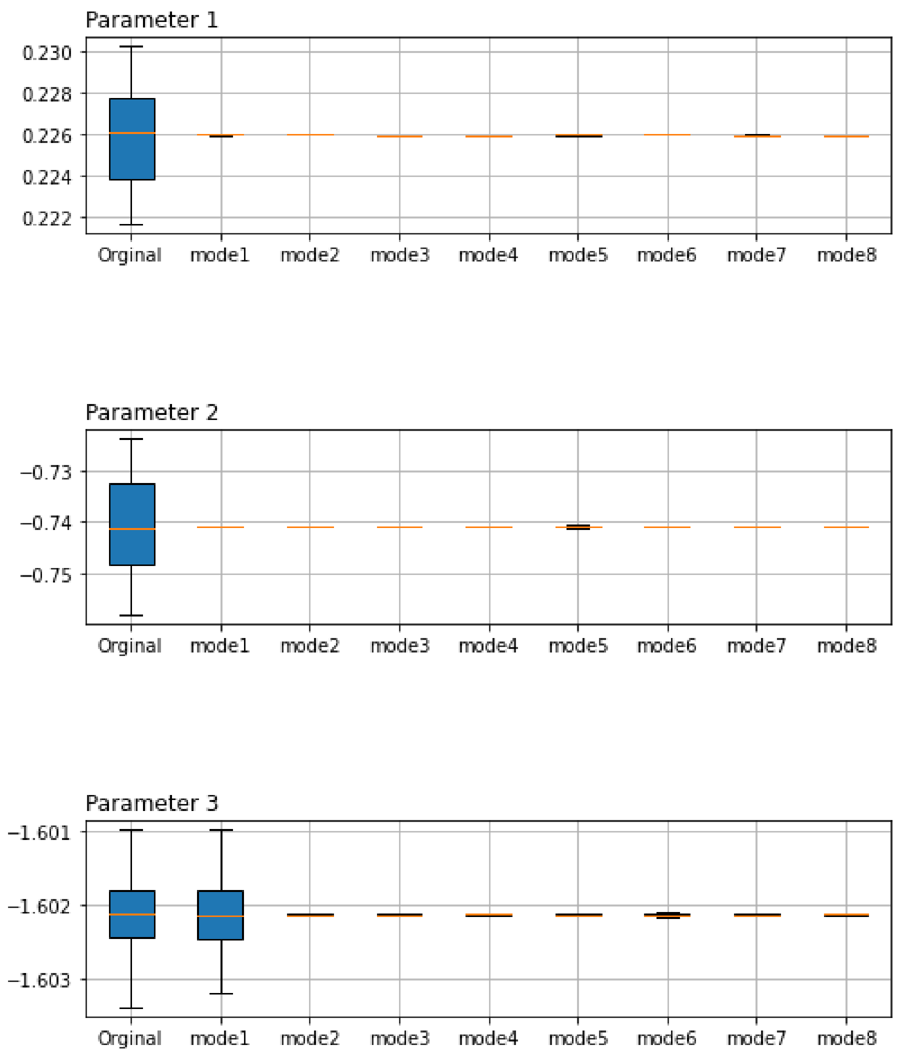

Figure B.3: Boxplot of each parameter in the original samples as well as samples after implementing 8 types. (P.S. mode'in the plot corresponds to 'type' in the thesis)

Case study 5: Bayesian probit model + Swiss dataset

Type	θ_1 VRR	θ_2 VRR	θ_3 VRR	θ_4 VRR	θ_5 VRR
type 1	9.486e-03	1.778e-02	1.476e-02	3.951e-02	5.378e-03
type 2	6.252e-03	1.511e-02	1.211e-02	1.414e-02	3.801e-03
type 3	3.662e-03	8.588e-03	7.191e-03	4.388e-03	7.884e-04
type 4	1.710e-03	1.512e-03	5.340e-04	8.313e-04	4.303e-04
type 5	9.998e-01	9.997e-01	9.998e-01	9.997e-01	9.997e-01
type 6	2.780e-01	2.492e-01	2.396e-01	1.221e-02	6.063e-01
type 7	5.531e-05	1.235e-04	1.441e-04	5.766e-05	6.440e-05
type 8	5.531e-05	1.235e-04	1.441e-04	5.766e-05	6.440e-05

Table B.10: Table shows the variance reduction ratio (VRR) of each type in sampling the posterior distribution of each parameter.

Type	*Number of Coefficients*	*Running time (s)*
type 1	5	7.642e-03
type 2	10	9.307e-03
type 3	14	1.183e-02
type 4	18	1.536e-02
type 5	6	7.780e-03
type 6	15	1.396e-02
type 7	20	1.662e-02
type 8	30	2.393e-02

Table B.11: Tables shows the number of coefficients that needed to be estimated and running time (apert from the original sampling) of each type in the generalized zero variance method. The original sampling spends 2148.023 seconds

Type	θ_1 ME	θ_2 ME	θ_3 ME	θ_4 ME	θ_5 ME
type 1	1.054e+02	5.625e+01	6.773e+01	2.531e+01	1.859e+02
type 2	1.600e+02	6.619e+01	8.260e+01	7.073e+01	2.631e+02
type 3	2.730e+02	1.164e+02	1.391e+02	2.279e+02	1.268e+03
type 4	5.849e+02	6.614e+02	1.873e+03	1.203e+03	2.324e+03
type 5	1.000e+00	1.000e+00	1.000e+00	1.000e+00	1.000e+00
type 6	3.597e+00	4.013e+00	4.173e+00	8.192e+01	1.649e+00
type 7	1.808e+04	8.098e+03	6.941e+03	1.734e+04	1.553e+04
type 8	2.914e+04	1.267e+04	1.048e+04	2.385e+04	4.353e+04

Table B.12: Table shows the measuring efficiency (ME) of each type in sampling the posterior distribution of each parameter.

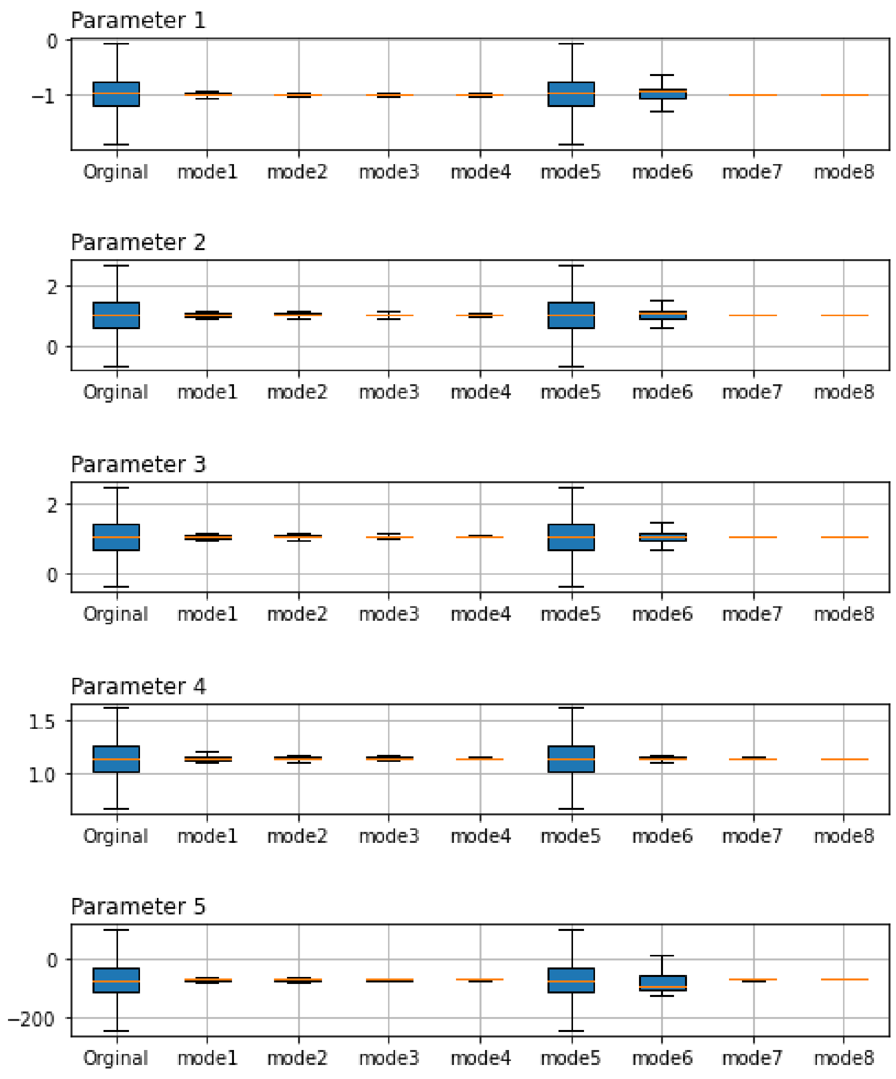

Figure B.4: Boxplot of each parameter in the original samples as well as samples after implementing 8 types. (P.S. mode'in the plot corresponds to 'type' in the thesis)

Appendix C

Colophon

This document was set in the UCL Overleaf template using pdfLaTeX. Word count is 10868 in overleaf and approximates 12000 in ’Word’.

Bibliography

[1] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

[2] Dirk P Kroese, Tim Brereton, Thomas Taimre, and Zdravko I Botev. Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):386–392, 2014.

[3] Marie-Estelle Demory, Pier Luigi Vidale, Malcolm J Roberts, Paul Berrisford, Jane Strachan, Reinhard Schiemann, and Matthew S Mizielinski. The role of horizontal resolution in simulating drivers of the global hydrological cycle. *Climate dynamics*, 42(7-8):2201–2225, 2014.

[4] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[5] Jeffrey S Rosenthal. A review of asymptotic convergence for general state space markov chains. *Far East J. Theor. Stat*, 5(1):37–50, 2001.

[6] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[7] Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992.

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

[9] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[10] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[11] Alessandro Barp, François-Xavier Briol, Anthony D Kennedy, and Mark Girolami. Geometry and dynamics for markov chain monte carlo. *Annual Review of Statistics and Its Application*, 5:451–471, 2018.

[12] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[13] Jesús A Izaguirre and Scott S Hampton. Shadow hybrid monte carlo: an efficient propagator in phase space of macromolecules. *Journal of Computational Physics*, 200(2):581–604, 2004.

[14] Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

[15] JEAN-MARIE CORNUET, JEAN-MICHEL MARIN, Antonietta Mira, and Christian P Robert. Adaptive multiple importance sampling. *Scandinavian Journal of Statistics*, 39(4):798–812, 2012.

[16] Subharup Guha and Steven N MacEachern. Generalized poststratification and importance sampling for subsampled markov chain monte carlo estimation. *Journal of the American Statistical Association*, 101(475):1175–1184, 2006.

[17] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[18] Anne Philippe and Christian P Robert. Riemann sums for mcmc estimation and convergence monitoring. *Statistics and Computing*, 11(2):103–115, 2001.

[19] Peter J Green, Krzysztof Łatuszyński, Marcelo Pereyra, and Christian P Robert. Bayesian computation: a perspective on the current state, and sampling backwards and forwards. *arXiv preprint arXiv:1502.01148*, 2015.

[20] Yves F Atchadé and François Perron. Improving on the independent metropolis-hastings algorithm. *Statistica Sinica*, pages 3–18, 2005.

[21] Hugo Hammer and Håkon Tjelmeland. Control variates for the metropolis–hastings algorithm. *Scandinavian Journal of Statistics*, 35(3):400–414, 2008.

[22] Shijing Si, Chris Oates, Andrew B Duncan, Lawrence Carin, François-Xavier Briol, et al. Scalable control variates for monte carlo methods via stochastic optimization. *arXiv preprint arXiv:2006.07487*, 2020.

[23] Charles Stein. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the sixth Berkeley symposium on mathematical statistics and probability, volume 2: Probability theory*, pages 583–602. University of California Press, 1972.

[24] Roland Assaraf and Michel Caffarel. Zero-variance principle for monte carlo algorithms. *Physical Review Letters*, 83(23):4682–4685, Dec 1999.

[25] Jackson Gorham. *Measuring sample quality with Stein's method.* Stanford University, 2017.

[26] Chris J. Oates, Mark Girolami, and Nicolas Chopin. Control functionals for monte carlo integration, 2016.

[27] Ruosi Wan, Mingjun Zhong, Haoyi Xiong, and Zhanxing Zhu. Neural control variates for monte carlo variance reduction. In *ECML/PKDD (2)*, pages 533–547, 2019.

[28] Antonietta Mira, Reza Solgi, and Daniele Imparato. Zero variance markov chain monte carlo for bayesian estimators. *Statistics and Computing*, 23(5):653–662, Jul 2012.

[29] Leah F South, Toni Karvonen, Chris Nemeth, Mark Girolami, Chris Oates, et al. Semi-exact control functionals from sard's method. *arXiv preprint arXiv:2002.00033*, 2020.

[30] Theodore Papamarkou, Antonietta Mira, and Mark Girolami. Zero variance differential geometric markov chain monte carlo algorithms. *Bayesian Analysis*, 9(1):97–128, 2014.

[31] Petros Dellaportas and Ioannis Kontoyiannis. Control variates for estimation based on reversible markov chain monte carlo samplers. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 74(1):133–161, 2012.

[32] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd ed. edition, 2004.

[33] `https://people.duke.edu/~ccc14/sta-663/PyStan.html`, note = Accessed September 2, 2021, title = Computational Statistics in Python.

[34] Bernhard Flury. *Multivariate statistics: a practical approach*. Chapman & Hall, Ltd., 1988.

[35] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1):1–32, 2017.

[36] Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo, 2011.

[37] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in python using pymc3. *PeerJ Computer Science*, 2:e55, 2016.

[38] Charles J. Geyer. Practical markov chain monte carlo. *Statistical Science*, 7(4):473–483, 1992.