A Mathematical formulation of the Monte Carlo method*

Hiroshi SUGITA

Department of Mathematics, Graduate School of Science, Osaka University

1 Introduction

Although admitting that the Monte Carlo method has been producing practical results in many fields, a number of researchers have the following serious suspicion about it.

> *The Monte Carlo method needs random numbers. But since no computer program can generate them, we execute a Monte Carlo method using a pseudorandom number generated by a computer program. Of course, such an expedient can never have any mathematical justification.*

As a matter of fact, this suspicion is a misunderstanding caused by prejudice. In this article, we propose a proper mathematical formulation of the Monte Carlo method, which resolves this suspicion.

The mathematical definitions of the notions of random number and pseudorandom number have been known for quite some time. Indeed, the notion of random number was defined in terms of computational complexity by Kolmogorov and others in 1960's ([3, 4, 6]), while the notion of pseudorandom number was defined in the context of cryptography by Blum and others in 1980's ([1, 2, 14]). But unfortunately, those definitions have been thought to be useless until now in understanding and executing the Monte Carlo method. It is not because their definitions are improper, but because our way of thinking about the Monte Carlo method has been improper. In this article, we propose a mathematical formulation of the Monte Carlo method which is based on and theoretically compatible with those definitions of random number and pseudorandom number.

Under our formulation, as a result, we see the following.

> *The Monte Carlo method may not need random numbers; pseudorandom numbers may suffice. As a matter of fact, for the purpose of the Monte Carlo integration, there exist pseudorandom numbers which can serve as complete substitutes for random numbers.*

The Monte Carlo integration is a numerical integration method making use of the law of large numbers. Since most of important applications of the Monte Carlo method are actually Monte Carlo integrations, the above fact is very significant.

---

* Ver.20151111 / This article is a digest of [11]. For details, see the full version [11].

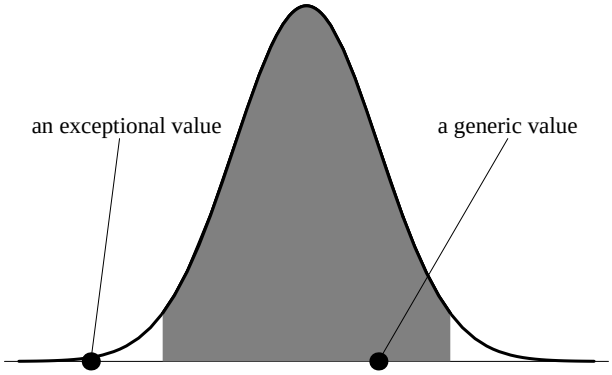
2 Overview

The *Monte Carlo method* is a numerical method to solve mathematical problems by computer-aided sampling of random variables. For each individual problem, we set up a probability space $(\Omega, \mathcal{F}, P)$ — in this article, for simplicity, we assume it to be a probability space of finite coin tosses

$$(\{0,1\}^L,\ 2^{\{0,1\}^L},\ P_L = \text{the uniform probability measure}), \quad L \in \mathbb{N}^+,$$

— and a random variable S defined on it; $S : \{0,1\}^L \to \mathbb{R}$. We evaluate $S(\omega)$ by a computer for some chosen $\omega \in \{0,1\}^L$, which procedure is called *sampling*.

Figure 1: Distribution and generic value of S (Conceptual figure)



The Monte Carlo method is a kind of *gambling* as its name indicates. The aim of the player, say Alice, is to get a *generic value* (Figure 1) — a typical value or not an exceptional value — of S by sampling (§ 3.1). Suppose that she has no idea about which ω she should choose to get a generic value of S. She chooses an $\omega \in \{0,1\}^L$ *of her own will*, realizing the risk of getting an exceptional value of S. Her risk is measured by the probability $P_L(A)$ of a set

$$A := \left\{ \omega \in \{0,1\}^L \,\middle|\, S(\omega) \text{ is an exceptional value of } S \right\}.$$

Since S seldom takes exceptional values, we have $P_L(A) \ll 1$,[†1] and hence, Alice may well think that she will almost certainly get a generic value of S. Of course she will, when L is small. But when $L \gg 1$, say $L = 10^8$, it is not sure that she will even though $P_L(A) \ll 1$.

In case $L \gg 1$, how Alice chooses an $\omega \in \{0,1\}^L$ comes into question. Indeed, to specify an $\omega \in \{0,1\}^L$, when $L = 10^8$ for example, she cannot help using a computer because of its huge amount of information; ω is approximately a 12MByte data. Obviously, a 12MByte data is too huge for her to input directly from a keyboard to a computer. So, she needs some device. But whatever device she may use, those ω's$\in \{0,1\}^L$ she can choose of her own will are very limited (§ 4). By this reason, even though $P_L(A) \ll 1$, we cannot say that Alice is able to get a generic value of S almost certainly. Therefore, to let

[†1] $a \ll b$ stands for "a is much less that b", while $a \gg b$ stands for "a is much greater than b ".

the risk evaluation $P_L(A)$ have a substantial meaning, it is natural to think that Alice needs an $\omega \in \{0,1\}^L$ which she cannot choose of her own will — namely, a *random number*.[†2]

On the other hand, S in an actual question must not be an arbitrary random variable, but one which represents some significant quantity. Thus S is very special among all the functions $\{0,1\}^L \to \mathbb{R}$. Therefore Alice may possibly be able to get a generic value of the special random variable S for a special ω — not a random number — which she can choose of her own will. It is pseudorandom generator that is planned to realize this idea.

A *pseudorandom generator* is a mapping which stretches short $\{0,1\}$-sequences into long $\{0,1\}$-sequences. For example, suppose that Alice uses a pseudorandom generator $g : \{0,1\}^n \to \{0,1\}^L$, $n < L$, to sample S. First, she chooses a *seed* $\omega' \in \{0,1\}^n$ of her own will. Here n should be so small that she may input ω' directly from a keyboard to a computer. Then the computer generates a *pseudorandom number* $g(\omega') \in \{0,1\}^L$ from her seed ω', and she finally gets a sample $S(g(\omega'))$ of S. As a result, she is now betting on *whether* $S(g(\omega'))$ *is a generic value of* S *or not*. Her risk is now measured by the probability $P_n(g(\omega') \in A)$. This probability naturally depends on g, but if there exists a pseudorandom generator g such that $P_n(g(\omega') \in A) \ll 1$ holds, Alice can actually get a generic value of S with high probability. Such a g is said to be *secure against* A (§ 5.2). The problem of sampling in a Monte Carlo method is solved by finding such a secure pseudorandom generator.

For a general random variable S, although there are many candidates, there is no pseudorandom generator which is proved to be secure against the set of those ω's $\in \{0,1\}^L$ that yield exceptional values of S (§ 5.3). However, if we restrict the use of pseudorandom generator to the *Monte Carlo integration*, i.e., if S in question is a sample mean of i.i.d.[†3] random variables, there exists a pseudorandom generator which is secure against the set of those ω's $\in \{0,1\}^L$ that yield exceptional values of S. Such a pseudorandom generator has already been used in practice for not too large Monte Carlo integrations (§ 6.2).

3 Monte Carlo method as gambling

Mathematical problems should be solved by sure methods, if it is possible. But some problems such as extremely complicated ones or those that lack a lot of information can only be solved stochastically. Those treated by the Monte Carlo method are such problems.

3.1 Player's aim

As a mathematical formulation, it is proper to think the Monte Carlo method as stochastic game, i.e., gambling. The aim of the player, Alice, is to get a generic value of a given random variable. (The mathematical problem in question is assumed to be solved by a generic value of the random variable. See examples below.) Of course, Alice has a risk to get an exceptional value, which risk should be measured in terms of probability. The following is a very small example of the Monte Carlo method (without computer).

[†2] By this reason, some people use physically generated random numbers. But this article deals with only solutions by pseudorandom generators.

[†3] i.i.d. stands for *i*ndependently *i*dentically *d*istributed.

Example 1 An urn contains 100 balls, 99 of which are numbered r and one of which is numbered $r+1$. Alice draws a ball from the urn, and guesses the number r to be the number of her ball. The probability that she fails to guess the number r correctly is $1/100$.

If we state this example in terms of gambling, it goes that "Alice wins if she draws a generic ball, i.e., a ball numbered r, and loses otherwise. The probability that she loses is $1/100$."

In general, the player cannot know whether the aim has been attained or not even after the sampling. Indeed, in the above example, although the risk is measured, Alice cannot tell if her guess is correct, even after she draws a ball.[†4]

There are some cases where exceptional values are needed. For example, we often seek for the minimum value of a complicated random variable X by a Monte Carlo method. In such a case, we define another random variable S so that a generic value of S is an exceptional value of X. Look at the following example.

Example 2 Suppose that $\Pr(X < c) = 1/10000$. So X can be less than c, but the probability is very small. Take a sequence $\{X_k\}_{k=1}^{40000}$ of independent copies of X, and define $S := \min_{1\le k\le 40000} X_k$. Then we have[†5]

$$\Pr(S < c) = 1 - \left(1 - \frac{1}{10000}\right)^{40000} \approx 1 - e^{-4} = 0.981\ldots.$$

Thus S takes a value less than c with high probability 0.98.

3.2 An exercise

To implement Example 1, we need only an urn and 100 numbered balls and nothing else. But actual Monte Carlo methods are implemented on such large scales that we need high-powered computers.

In this article, we are going to solve the following exercise by a Monte Carlo method.

> **Exercise** When we toss a coin 100 times, what is the probability p that Heads comes up at least 6 times in succession?

We apply the interval estimation in mathematical statistics. Repeat independent trials of "100 coin tosses" N times, and let S_N be the number of the occurrences of "Heads comes up at least 6 times in succession" among the trials. Then by the law of large numbers, the sample mean S_N/N is a good estimator for p when N is large. More concretely;

Example 3 Let $N := 10^6 = 1,000,000$. Then the mean and the variance of $S_{10^6}/10^6$ are

$$\mathbf{E}\left[\frac{S_{10^6}}{10^6}\right] = p, \quad \mathbf{V}\left[\frac{S_{10^6}}{10^6}\right] = \frac{p(1-p)}{10^6} \le \frac{1}{4\cdot 10^6},$$

[†4] Many selections of our life are certainly gambles. It is often the case where we do not know whether the selection was correct or not ....

[†5] $x \approx y$ means that x and y are approximately equal to each other.

respectively. Hence by *Chebyshev's inequality*,

$$\Pr\left(\left|\frac{S_{10^6}}{10^6} - p\right| \geq \frac{1}{200}\right) \leq \frac{1}{4 \cdot 10^6} \cdot 200^2 = \frac{1}{100} \tag{1}$$

holds. In other words, a generic value of $S_{10^6}/10^6$ is an approximate value of p.

In Example 3, we may think that the inequality (1) measures the risk.

Of course, we do not toss a coin $100 \times 10^6 = 10^8$ times, instead we use a computer. In order to formulate things mathematically, we realize the random variable S_{10^6} of Example 3 on the coin tossing probability space $(\Omega := \{0,1\}^{10^8}, 2^\Omega, P_{10^8})$ as follows. We first define a function $X : \{0,1\}^{100} \to \{0,1\}$ by

$$X(\xi_1, \ldots, \xi_{100}) := \max_{1 \leq l \leq 100-5} \prod_{i=l}^{l+5} \xi_i, \quad (\xi_1, \ldots, \xi_{100}) \in \{0,1\}^{100}.$$

This means that $X = 1$ if there are 6 successive 1's in $(\xi_1, \ldots, \xi_{100})$ and $X = 0$ otherwise. Next we define $X_k : \{0,1\}^{10^8} \to \{0,1\}$, $k = 1, 2, \ldots, 10^6$, by

$$X_k(\omega) := X(\omega_{100(k-1)+1}, \ldots, \omega_{100k}), \quad \omega = (\omega_1, \ldots, \omega_{10^8}) \in \{0,1\}^{10^8},$$

and $S_{10^6} : \{0,1\}^{10^8} \to \mathbb{Z}$ by

$$S_{10^6}(\omega) := \sum_{k=1}^{10^6} X_k(\omega), \quad \omega \in \{0,1\}^{10^8}.$$

Defining A_0 of ω's that yield exceptional values of S_{10^6} by

$$A_0 := \left\{\omega \in \{0,1\}^{10^8} \;\middle|\; \left|\frac{S_{10^6}(\omega)}{10^6} - p\right| \geq \frac{1}{200}\right\}, \tag{2}$$

we have $P_{10^8}(A_0) \leq 1/100$ according to (1).

Then we can regard Example 3 as a gamble in the following way.

Example 4 Alice chooses an $\omega \in \{0,1\}^{10^8}$. If $\omega \notin A_0$ holds, she wins, if $\omega \in A_0$, she loses. The probability that she loses is less than or equal to $1/100$.

4 Problem of random number

There are no theoretical differences between Example 1 and Example 4 except scales. But the difference of scales yields an essential difference in practice.

Now, suppose that Alice is going to choose an $\omega \in \{0,1\}^{10^8}$ to play the gamble of Example 4. But as a matter of fact, those ω's $\in \{0,1\}^{10^8}$ she can choose of her own will are very few and hence limited. Consequently, even though $P_{10^8}(A_0) \ll 1$, it is not necessarily easy for her to win the game.

Let us look more closely at the situation. Since each element of $\{0,1\}^{10^8}$ is a very large data, i.e., about 12MByte, Alice cannot help using a computer to choose an $\omega \in$

$\{0,1\}^{10^8}$. But it is quite impossible to input it directly from a keyboard to the computer. Suppose that Alice can input at most 1,000 bit data directly from the keyboard, and that a certain computer program generates an element of $\{0,1\}^{10^8}$ from her input.[†6] Then the number of those ω's $\in \{0,1\}^{10^8}$ that she can choose is at most $2^{1,001}$. (This is because the number of all the l bit data is 2^l, and hence the number of all data of at most l bit is $2^0 + 2^1 + 2^2 + \cdots + 2^l = 2^{l+1} - 1$.) Since the number of all the elements of $\{0,1\}^{10^8}$ is 2^{10^8}, we see how few those ω's $\in \{0,1\}^{10^8}$ she can choose are. Even if she were able to input at most $10^8 - 10$ bit data, the number of ω's $\in \{0,1\}^{10^8}$ she can choose would be at most 2^{10^8-9}, which is only $1/512$ of all the ω's $\in \{0,1\}^{10^8}$. In other words, at least $511/512$ of them require more than $10^8 - 9$ bit input to be specified.

If an $\omega \in \{0,1\}^L$, $L \gg 1$, requires an input which is almost as long as ω itself to be specified, it is called a *random number*.[†7] To specify a random number, there is no more efficient way than inputting it as it is. When $L \gg 1$, most of elements of $\{0,1\}^L$ are random numbers.

The risk evaluation $P_{10^8}(A_0) \leq 1/100$ of Example 4 assumes that Alice can choose any $\omega \in \{0,1\}^{10^8}$ with equal probability. This means that she should choose it among random numbers, because they account for nearly all sequences. This is the reason why random number is needed for the Monte Carlo method. However, although there are so many random numbers, she cannot choose any one of them of her own will. This is the most essential problem of sampling in the Monte Carlo method.

5 Pseudorandom generator

Pseudorandom generator is a device to get generic values of random variables with high probability without using random numbers. An important property that pseudorandom generators should have is discussed here.

5.1 Definition and role

To play the gamble of Example 4, anyhow, Alice has to choose an $\omega \in \{0,1\}^{10^8}$. Let us suppose that she uses the most used device to do it, namely, a pseudorandom generator.

Definition 5 A function $g : \{0,1\}^n \to \{0,1\}^L$ is called a *pseudorandom generator* if $n < L$. The input $\omega' \in \{0,1\}^n$ of g is called a *seed*,[†8] and the output $g(\omega') \in \{0,1\}^L$ a *pseudorandom number*.

To produce a pseudorandom number, we need to choose a seed $\omega' \in \{0,1\}^n$ of $g : \{0,1\}^n \to \{0,1\}^L$, which procedure is called *initialization*.[†9] For practical use, n should be so small that we may input any seed $\omega' \in \{0,1\}^n$ directly from a keyboard, and the program of the function g should work sufficiently fast.

[†6] Such a computer program is called a pseudorandom generator.

[†7] In the IT terminology, if an $\omega \in \{0,1\}^L$ can be specified by a shorter input $\omega' \in \{0,1\}^n$, $L > n$, we say that ω is compressed into ω'. Thus a random number is an incompressible element of $\{0,1\}^L$.

[†8] We also call it an *initial value*.

[†9] It is also called *randomization*.

Example 6 In Example 4, suppose that Alice uses a pseudorandom generator $g : \{0,1\}^{238} \to \{0,1\}^{10^8}$, for instance.[†10] She chooses a seed $\omega' \in \{0,1\}^{238}$ of g and inputs it from a keyboard to a computer. Since ω' is only a 238 bit data ($\approx$ 30 letters of alphabet), it is easy to input from a keyboard. Then the computer produces $S_{10^6}(g(\omega'))$.

The reason why Alice uses a pseudorandom generator is because her input $\omega \in \{0,1\}^{10^8}$ is too large. If it is short enough, a pseudorandom generator is not necessary. For example, when drawing a ball from the urn in Example 1, who on earth uses a pseudorandom generator ?

5.2 Security

Let us continue to consider the case of Example 6. Alice can choose any seed $\omega' \in \{0,1\}^{238}$ of the pseudorandom generator g freely of her own will. Her risk is measured by

$$P_{238}\left(\left|\frac{S_{10^6}(g(\omega'))}{10^6} - p\right| \geq \frac{1}{200}\right), \tag{3}$$

which we need to calculate. Of course, the probability (3) depends on g. If this probability — i.e., the probability that her sample $S_{10^6}(g(\omega'))$ is an exceptional value of S — is large, then it is difficult for her to win the game, which is not desirable.

So we give the following (somewhat vague) definition; we say that a pseudorandom generator $g : \{0,1\}^n \to \{0,1\}^L$, $n < L$, is *secure against a set* $A \subset \{0,1\}^L$ if it holds that

$$P_L(\omega \in A) \approx P_n(g(\omega') \in A).$$

In Example 6, if $g : \{0,1\}^{238} \to \{0,1\}^{10^8}$ is secure against A_0 of (2), for the majority of the seeds $\omega' \in \{0,1\}^{238}$, which Alice can choose of her own will, the samples $S(g(\omega'))$ will be generic values of S. In this case, random numbers are not necessary. In other words, in sampling a value of S, using g does not make Alice's risk big, and so g is said to be secure. The problem of sampling in each Monte Carlo method — i.e., the problem of random number — will be resolved by finding a suitable secure pseudorandom generator.

In general, such a pseudorandom generator that is secure against very many sets A's is desirable. But there is no pseudorandom generator that is against every set A. Indeed, for a given pseudorandom generator $g : \{0,1\}^n \to \{0,1\}^L$, set

$$A_g := g(\{0,1\}^n) \subset \{0,1\}^L.$$

Then we have $P_L(\omega \in A_g) \leq 2^{n-L}$ and $P_n(g(\omega') \in A_g) = 1$, which means that g is not secure against A_g. Thus, when we consider the security of pseudorandom generator, we must restrict a class of sets A's.

5.3 Computationally secure pseudorandom generator

Suppose $L \gg 1$, e.g., $L = 10^8$. For each set $A \subset \{0,1\}^L$, let us think about how to judge if a pseudorandom generator g is secure against it. To do this, first of all, we must write

[†10] The origin of the number 238 will soon be clear in Example 9.

a program which judges if a given $\omega \in \{0,1\}^L$ belongs to A. Since the number of all the sets $A \subset \{0,1\}^L$ is 2^{2^L}, the same number of such programs are needed. Then by the same argument in § 4, we know there are very many sets A's for which we need about 2^L bit long programs. Obviously, such long programs cannot be realized by computers in practice. Consequently, it is practically impossible to judge if $\omega \in A$ for most of A's$\subset \{0,1\}^L$.

A pseudorandom generator g needs to be secure against only A for which $\omega \in A$ is practically judge-able by small amount of computations. Such a g is said to be *computationally secure* (or *cryptographically secure*). In cryptography, however, the security of pseudorandom generator is not defined in terms of space complexity (i.e., length of program, as is discussed above), but it is defined in terms of time complexity (i.e., CPU time).[†11]

In Example 6, if the pseudorandom generator $g : \{0,1\}^{238} \to \{0,1\}^{10^8}$ is computationally secure, the distributions of $S_{10^6}(\omega)$ and $S_{10^6}(g(\omega'))$ will be so close to each other. Indeed, the fact that the function S_{10^6} can be computed in practice means that for any $c_1 < c_2 \in \mathbb{N}$, the set $A(c_1, c_2) := \{\omega \,|\, c_1 \le S_{10^6}(\omega) \le c_2\}$ is a computationally judge-able set. Therefore whatever p' may be, the computational security of g implies that

$$P_{10^8}\left(\left|\frac{S_{10^6}(\omega)}{10^6} - p'\right| \ge \frac{1}{200}\right) \approx P_{238}\left(\left|\frac{S_{10^6}(g(\omega'))}{10^6} - p'\right| \ge \frac{1}{200}\right).$$

Computationally secure pseudorandom generator is, theoretically speaking, the most complete multi-purpose pseudorandom generator. But, its existence has not been proved yet, which is one of the hardest problems in computer science. Moreover, the notion of computational security is, rigorously speaking, an asymptotic property, and hence, it is not clear that a computationally secure pseudorandom generator is surely useful for a practical use.

6 Monte Carlo integration

Let X be a function of m coin tosses, i.e., $X : \{0,1\}^m \to \mathbb{R}$, and let us consider to calculate the mean

$$\mathbf{E}[X] = \frac{1}{2^m} \sum_{\xi \in \{0,1\}^m} X(\xi)$$

of X numerically. When m is small, we can directly calculate the finite sum of the right hand side. But when m is large, e.g., $m = 100$, the direct calculation becomes impossible in practice because of the huge amount of computation. In such a case, we estimate the mean of X applying the law of large numbers, which is called the *Monte Carlo integration* (Example 3). Most of scientific Monte Carlo methods aim at calculating some characteristics of distributions of random variables, which are actually Monte Carlo integrations.

6.1 i.i.d.-sampling

If we formulate Example 3 in a general setting, it goes as follows. Let $\{X_k\}_{k=1}^N$ be a sequence of independent copies of $X : \{0,1\}^m \to \mathbb{R}$, and S_N be their sum. Namely, $\{X_k\}_{k=1}^N$

[†11] Since a short program can take time if it includes many loops, it is practically significant to define the security in terms of time complexity.

and S_N are functions of Nm coin tosses, which are written down as

$$X_k(\omega) := X(\omega_k), \quad \omega_k \in \{0,1\}^m, \quad \omega = (\omega_1, \ldots, \omega_N) \in \{0,1\}^{Nm}, \tag{4}$$

$$S_N(\omega) := \sum_{k=1}^{N} X_k(\omega). \tag{5}$$

Then if N is large enough, a generic value of S_N/N becomes an approximated value of $\mathbf{E}[X]$ by the law of large numbers. The estimation of the mean of X by sampling the random variable S_N/N is called the *i.i.d.-sampling*.

Since the means of S_N/N and X (i.e., the integrations with respect to P_{Nm} and P_m, respectively) are the same (i.e., $\mathbf{E}[S_N/N] = \mathbf{E}[X]$), and their variances satisfy $\mathbf{V}[S_N/N] = \mathbf{V}[X]/N$, we have the following inequality due to Chebyshev ;[†12]

$$P_{Nm}\left(\left| \frac{S_N(\omega)}{N} - \mathbf{E}[X] \right| \geq \delta \right) \leq \frac{\mathbf{V}[X]}{N\delta^2}. \tag{6}$$

This is the risk measurement of the following gamble; when Alice chooses an $\omega \in \{0,1\}^{Nm}$, she wins if $\omega \notin A_1$, and she loses if $\omega \in A_1$, where

$$A_1 := \left\{ \omega \in \{0,1\}^{Nm} \;\middle|\; \left| \frac{S_N(\omega)}{N} - \mathbf{E}[X] \right| \geq \delta \right\}. \tag{7}$$

6.2 Random Weyl sampling

In the Monte Carlo integration, the object random variable S_N to sample has a very special form. Utilizing this fact, we can concretely construct a secure pseudorandom generator to sample it.

Before doing it, we need a couple of notations; let

$$D_m := \{\, i2^{-m} \,|\, i = 0, ..., 2^m - 1 \,\} \subset \mathbb{T}^1. \tag{8}$$

Let $\mathcal{B}_m$ be the algebra generated by the collection of sets $\mathcal{I}_m := \{[a,b) | a, b \in D_m\}$. Namely, each element of $\mathcal{B}_m$ is a finite union of some elements of $\mathcal{I}_m$. Let $P_{(m)}$ be the uniform probability measure on D_m. For each $m \geq 1$ and each $x \in \mathbb{T}^1$, let

$$\lfloor x \rfloor_m := \lfloor 2^m x \rfloor / 2^m \in D_m.$$

Definition 7 (cf. [7, 13]) Let $j \in \mathbb{N}^+$, and set

$$Z_k(\omega') := \lfloor x + k\alpha \rfloor_m \in D_m, \quad \omega' = (x, \alpha) \in D_{m+j} \times D_{m+j}, \quad k = 1, 2, 3, \ldots, 2^{j+1}.$$

Then we define a pseudorandom generator $g : \{0,1\}^{2m+2j} \to \{0,1\}^{Nm}$, $N \leq 2^{j+1}$, by

$$\begin{aligned} g(\omega') &:= (Z_1(\omega'), Z_2(\omega'), \ldots, Z_N(\omega')) \in (D_m)^N \cong \{0,1\}^{Nm}, \\ \omega' &= (x, \alpha) \in D_{m+j} \times D_{m+j} \cong \{0,1\}^{2m+2j}. \end{aligned} \tag{9}$$

The numerical integration method using the pseudorandom generator (9) for the sampling of S_N is called the *random Weyl sampling* (abbreviated as RWS).[†13]

[†12] In general, as $\mathbf{E}[X]$ is unknown, $\mathbf{V}[X]$ is unknown, too. In this sense, this risk measurement is not complete. But under some circumstances, an upper bound $\mathbf{V}[X]$ can be obtained (e.g., Example 3), the risk measurement then becomes complete.

[†13] RWS presented here is slightly improved from the one introduced by [7, 13].

Theorem 8 *The pseudorandom generator $g : \{0,1\}^{2m+2j} \to \{0,1\}^{Nm}$ of (9) satisfies that for S_N of (5),*

$$\begin{aligned} \mathbf{E}[S_N(g(\omega'))] &= \mathbf{E}[S_N(\omega)] \,(= N\mathbf{E}[X]), \\ \mathbf{V}[S_N(g(\omega'))] &= \mathbf{V}[S_N(\omega)] \,(= N\mathbf{V}[X]). \end{aligned}$$

Here ω' and ω are assumed to be distributed uniformly in $\{0,1\}^{2m+2j}$ and in $\{0,1\}^{Nm}$, respectively. From this, as is seen in (6), Chebyshev's inequality

$$P_{2m+2j}(g(\omega') \in A_1) \;=\; P_{2m+2j}\left(\left|\frac{S_N(g(\omega'))}{N} - \mathbf{E}[X]\right| \geq \delta\right) \;\leq\; \frac{\mathbf{V}[X]}{N\delta^2}$$

follows. In this sense, g is secure against A_1 of (7).

Proof. Step 1. (cf. [7, 13]) Under the uniform (direct product) probability measure $P_{(m+j)} \otimes P_{(m+j)}$ on $D_{m+j} \times D_{m+j}$, we show that $\{Z_k\}_{k=1}^{2^{j+1}}$ are pairwise independent, and that each Z_k is distributed uniformly in D_m. To this end, we prove that for any $\mathcal{B}_m$-measurable functions F, $G : \mathbb{T}^1 \to \mathbb{R}$ and for any $1 \leq k < k' \leq 2^{j+1}$, it holds that

$$\mathbf{E}[F(Z_{k'})G(Z_k)] = \int_0^1 F(t)dt \int_0^1 G(s)ds. \tag{10}$$

Here $\mathbf{E}$ denotes the mean under $P_{(m+j)} \otimes P_{(m+j)}$.

By the $\mathcal{B}_m$-measurability of F and G, we have

$$\begin{aligned} \mathbf{E}[F(Z_{k'})G(Z_k)] &= \frac{1}{2^{2m+2j}} \sum_{q=1}^{2^{m+j}} \sum_{p=1}^{2^{m+j}} F\left(\frac{p}{2^{m+j}} + \frac{k'q}{2^{m+j}}\right) G\left(\frac{p}{2^{m+j}} + \frac{kq}{2^{m+j}}\right) \\ &= \frac{1}{2^{2m+2j}} \sum_{q=1}^{2^{m+j}} \sum_{p=1+kq}^{2^{m+j}+kq} F\left(\frac{p}{2^{m+j}} + \frac{(k'-k)q}{2^{m+j}}\right) G\left(\frac{p}{2^{m+j}}\right) \\ &= \frac{1}{2^{2m+2j}} \sum_{q=1}^{2^{m+j}} \sum_{p=1}^{2^{m+j}} F\left(\frac{p}{2^{m+j}} + \frac{(k'-k)q}{2^{m+j}}\right) G\left(\frac{p}{2^{m+j}}\right). \end{aligned} \tag{11}$$

Now, let us assume that $0 < k' - k = 2^i l \leq 2^{j+1} - 1$, where $0 \leq i \leq j$ and l is an odd number. Then we have

$$\frac{1}{2^{m+j}} \sum_{q=1}^{2^{m+j}} F\left(\frac{p}{2^{m+j}} + \frac{(k'-k)q}{2^{m+j}}\right) = \frac{1}{2^{m+j}} \sum_{q=1}^{2^{m+j}} F\left(\frac{p}{2^{m+j}} + \frac{lq}{2^{m+j-i}}\right). \tag{12}$$

For each $r = 1, 2, 3, \ldots, 2^{m+j-i}$, there exists a q_r such that $lq_r \equiv r \,(\mathrm{mod}\, 2^{m+j-i})$. Since l is odd,

$$\begin{aligned} &\#\{1 \leq q \leq 2^{m+j} \,|\, lq \equiv r \,(\mathrm{mod}\, 2^{m+j-i})\} \\ &\quad= \#\{1 \leq q \leq 2^{m+j} \,|\, lq \equiv lq_r \,(\mathrm{mod}\, 2^{m+j-i})\} \\ &\quad= \#\{1 \leq q \leq 2^{m+j} \,|\, l(q - q_r) \equiv 0 \,(\mathrm{mod}\, 2^{m+j-i})\} \\ &\quad= \#\{1 \leq q \leq 2^{m+j} \,|\, q \equiv q_r \,(\mathrm{mod}\, 2^{m+j-i})\} \\ &\quad= 2^i. \end{aligned}$$

From this, it follows that

$$\begin{aligned}\frac{1}{2^{m+j}}\sum_{q=1}^{2^{m+j}} F\left(\frac{p}{2^{m+j}}+\frac{lq}{2^{m+j-i}}\right) &= \frac{1}{2^{m+j-i}}\sum_{r=1}^{2^{m+j-i}} F\left(\frac{p}{2^{m+j}}+\frac{r}{2^{m+j-i}}\right)\\ &= \frac{1}{2^{m+j-i}}\sum_{r=1}^{2^{m+j-i}} F\left(\frac{r}{2^{m+j-i}}\right)\\ &= \int_0^1 F(t)dt. \qquad (13)\end{aligned}$$

By (11), (12) and (13), we see

$$\begin{aligned}\mathbf{E}[F(Z_{k'})G(Z_k)] &= \frac{1}{2^{m+j}}\sum_{p=1}^{2^{m+j}}\left(\frac{1}{2^{m+j}}\sum_{q=1}^{2^{m+j}} F\left(\frac{p}{2^{m+j}}+\frac{(k'-k)q}{2^{m+j}}\right)\right)G\left(\frac{p}{2^{m+j}}\right)\\ &= \frac{1}{2^{m+j}}\sum_{p=1}^{2^{m+j}}\left(\frac{1}{2^{m+j}}\sum_{q=1}^{2^{m+j}} F\left(\frac{p}{2^{m+j}}+\frac{lq}{2^{m+j-i}}\right)\right)G\left(\frac{p}{2^{m+j}}\right)\\ &= \int_0^1 F(t)dt\cdot\frac{1}{2^{m+j}}\sum_{p=1}^{2^{m+j}} G\left(\frac{p}{2^{m+j}}\right) = \int_0^1 F(t)dt\int_0^1 G(s)ds.\end{aligned}$$

Thus (10) is proved.

Step 2. First, since each $Z_k(\omega')$ is distributed uniformly in $\{0,1\}^m$, we see

$$\mathbf{E}[S_N(g(\omega'))] = N\mathbf{E}[X].$$

Next, the pairwise independence implies that

$$\begin{aligned}\mathbf{V}[S_N(g(\omega'))] &= \mathbf{E}\left[\left(\sum_{k=1}^{N}(X(Z_k(\omega'))-\mathbf{E}[X])\right)^2\right]\\ &= \sum_{k=1}^{N}\sum_{k'=1}^{N}\mathbf{E}\left[(X(Z_k(\omega'))-\mathbf{E}[X])\,(X(Z_{k'}(\omega'))-\mathbf{E}[X])\right]\\ &= \sum_{k=1}^{N}\mathbf{E}\left[(X(Z_k(\omega'))-\mathbf{E}[X])^2\right]\\ &\quad +2\sum_{1\le k<k'\le N}\mathbf{E}\left[(X(Z_k(\omega'))-\mathbf{E}[X])\,(X(Z_{k'}(\omega'))-\mathbf{E}[X])\right]\\ &= N\mathbf{V}[X].\end{aligned}$$

Thus we know that g has the required properties. □

Example 9 Applying RWS, we can solve Exercise in § 3.2. In Example 6 (§ 5.1), let us use the pseudorandom generator $g : \{0,1\}^{238} \to \{0,1\}^{10^8}$ defined by (9) with $m = 100$, $N = 10^6$, and $j = 19$.[†14] Then the risk is measured by (cf. (3))

$$P_{238}\left(\left|\frac{S_{10^6}(g(\omega'))}{10^6}-p\right|\ge\frac{1}{200}\right)\le\frac{1}{100}. \qquad (14)$$

[†14] Here we have $2^{j+1} = 2^{20} > 10^6 = N$ and $2m + 2j = 238$. In a practical Monte Carlo integration, the sample size N is not determined in advance, but it is usually determined in doing numerical experiments. To be ready for such situations, it is a good idea to let j be a little on the big side.

Since Alice can easily choose any seed $\omega' \in \{0,1\}^{238}$ of her own will, she no longer needs a long random number. Here is a concrete example. Instead of her, the author chose the following seed $\omega' = (x, \alpha) \in D_{119} \times D_{119} \cong \{0,1\}^{238}$ written in dyadic expansion;

$$
\begin{aligned}
x \;=\;& \texttt{0.1110110101 1011101101 0100000011 0110101001 0101000100} \\
& \texttt{0101111101 1010000000 1010100011 0100011001 1101111101} \\
& \texttt{1101010011 111100100},\\
\alpha \;=\;& \texttt{0.1100000111 0111000100 0001101011 1001000001 0010001000} \\
& \texttt{1010101101 1110101110 0010010011 1000000011 0101000110} \\
& \texttt{0101110010 010111111}.
\end{aligned}
$$

Then the computer calculated as $S_{10^6}(g(\omega')) = 546,177$. In this case,

$$\frac{S_{10^6}(g(\omega'))}{10^6} \;=\; 0.546177$$

is the estimated value of the probability p.

Remark 10 When Alice executes RWS, we can advise her a little in choosing a seed $\omega' = (x, \alpha) \in \{0,1\}^{2m+2j}$. That is, she should not choose a particularly simple α. Indeed, if she chooses an extremely simple one, such as $\alpha = (0, 0, \ldots, 0) \in \{0,1\}^{m+j}$, the sampling will certainly end in failure.

Remark 11 In case $2m + 2j \gg 1$, the problem of random number again prevents Alice even from choosing a seed $\omega' \in \{0,1\}^{2m+2j}$ of RWS of her own will. In such a case, she is forced to choose it by an auxiliary pseudorandom generator $g' : \{0,1\}^n \to \{0,1\}^{2m+2j}$. Then we do not know whether or not the composite generator $g \circ g' : \{0,1\}^n \to \{0,1\}^{2m+2j} \to \{0,1\}^{Nm}$ is secure against A_1 of (7).

Example 12 Let us apply a pairwise independent sampling such as RWS to search the minimum value of the random variable X which was dealt with in Example 2. Suppose that $\Pr(X < c) = 1/10000$ and that $X'_1, X'_2, \ldots, X'_{40000}$ are pairwise independent copies of X. Then setting $S' := \sum_{k=1}^{40000} \mathbf{1}_{\{X'_k < c\}}$, we have

$$\mathbf{E}[S'] = 4, \quad \mathbf{V}[S'] = 40000\mathbf{V}[\mathbf{1}_{\{X'_k<c\}}] = 40000\left(1 - \frac{1}{10000}\right)\frac{1}{10000} \;<\; 4.$$

Therefore Chebyshev's inequality implies that

$$\Pr(S' \geq 1) \;\geq\; \Pr\left(|S' - 4| < 4\right) \;\geq\; 1 - \frac{4}{4^2} \;=\; \frac{3}{4}.$$

Thus the random variable

$$\min_{1 \leq k \leq 40000} X'_k$$

takes values less than c with probability at least $3/4$.

7 From viewpoint of mathematical statistics

7.1 Random sampling

We have formulated the Monte Carlo method as gambling and we have considered that the seed $\omega' \in \{0,1\}^n$ of a pseudorandom generator g is chosen by Alice of her own will. But from the viewpoint of mathematical statistics, this is not a good formulation, because sampling should be done randomly in order to guarantee the objectivity of the result. Indeed, in the case of RWS, as we mentioned in Remark 10, Alice can choose a bad seed on purpose, i.e., the result may depend on the player's will.

Of course, it is impossible to discuss the objectivity of sampling rigorously. We here simply assume that Heads or Tails of coin tosses do not depend on anyone's will. Then, for instance, when we choose a seed $\omega' \in \{0,1\}^n$, we toss a coin n times, record 1 if Heads comes up and 0 if Tails does at each coin toss, define ω' as the recorded $\{0,1\}$-sequence, and finally compute $S(g(\omega'))$, which completes an objective sampling. As a matter of fact, Example 9 was performed in this way.

This method cannot be used to choose a very long $\omega \in \{0,1\}^L$. The point is that the pseudorandom generator $g : \{0,1\}^n \to \{0,1\}^L$ makes the input shorter so that this method may become executable.

7.2 Test for pseudorandom generator

Many statistical tests for pseudorandom generators have been done in the following manner. Let $g : \{0,1\}^n \to \{0,1\}^L$ be a pseudorandom generator to test.

(1) Decide what test to do. (Test of run, Poker test, ...)

(2) For randomly chosen seeds $\omega' \in \{0,1\}^n$, generate pseudorandom numbers $g(\omega')$ by g, calculate the portion of rejected ones.

(3) If the portion of rejected ones is close to the significance level of the test, accept g, if it is much greater than the significance level, reject g.

In the above procedure, if we let A denote the rejection region of the test chosen at (1), what is done at (2) is the estimation of the probability $P_n(g(\omega') \in A)$. And at (3), we check if it is close to the significance level $P_L(\omega \in A)$. As a result, this test can be said to be a test for the security of g against A.

8 Concluding remarks

As is seen in this article, needless to say, secure pseudorandom generators are useful. Here we mention about the relation between random number and probability theory.

In Kolmogorov's modern probability theory, a random quantity is expressed as a random variable X, which is a function defined on a probability space, say $(\{0,1\}^L, 2^{\{0,1\}^L}, P_L)$, i.e., $X : \{0,1\}^L \to \mathbb{R}$. We think X random by the following interpretation; an $\omega \in \{0,1\}^L$ is randomly chosen and as a result $X(\omega)$ becomes random. But in probability theory, we

always deal with X as just a function and we never mind how ω is chosen. Since randomness does lie in the process how ω is chosen, probability theory does not mind what randomness is.

Suffering from Parkinson's disease in his later years, Kolmogorov devoted himself to the question "What is randomness?", which his probability theory had been avoiding. Finally, he answered the question, by establishing the notion of random number. Theoretically, we do not need the notion of random number in studying probability theory, but recognizing it brings us deep understanding of not only the Monte Carlo method but also probability theory itself. Let us explain it below.

According to Kolmogorov, studying randomness is equivalent to studying random number. The existence of random numbers becomes prominent, only when the sample space $\{0,1\}^L$ in question is huge. We therefore know that it is important to study the case $L \gg 1$. So, let us suppose $L \gg 1$. Then, we cannot choose any one of random numbers in $\{0,1\}^L$ of our own will, although they account for nearly all sequences. This means that assuming the uniform probability measure P_L implies that $\omega \in \{0,1\}^L$ is not assumed to be chosen by anyone's will, but by some method beyond man's will, i.e., at random. Thus the probability space $(\{0,1\}^L, 2^{\{0,1\}^L}, P_L)$ provides a framework to study randomness.

It is difficult to get any knowledge about individual random numbers. But since random numbers account for nearly all sequences, it is a good idea to study characteristic properties that nearly all sequences share. Such properties have been minutely studied in probability theory — properties described in various limit theorems, such as law of large numbers, central limit theorems, etc. Probably, limit theorems are the only mathematical formulation that enables us to investigate randomness concretely. This explains why limit theorems are so much studied in probability theory.

What is really amazing is that since a long time before the discovery of the notion of random number, the probabilists of great insight had recognized the importance of limit theorems and had made a lot of efforts to study them.

References

[1] L. Blum, M. Blum and M. Shub, A simple unpredictable pseudorandom number generator, *SIAM J. Comput.*, **15-2** (1986), 364–383.

[2] M. Blum and S. Macali, How to generate cryptographically strong sequences of pseudorandom bits, *SIAM J. on Computing*, vol. **13**, (1984) 850–864. A preliminary version appears in *Proceedings of the IEEE Foundations of Comput. Sci.* (1982), 112–117.

[3] G.J. Chaitin, Algorithmic information theory, *IBM J. Res. Develop.*, **21** (1977), 350–359.

[4] A.N. Kolmogorov, *Selected works of A. N. Kolmogorov. Vol. III. Information theory and the theory of algorithms*, Edited by A. N. Shiryayev [A. N. Shiryaev]. Translated from the 1987 Russian original by A. B. Sossinsky. Mathematics and its Applications (Soviet Series), 27. Kluwer Academic Publishers Group, Dordrecht, (1993) xxvi+275 pp.

[5] M. Luby, *Pseudorandomness and cryptographic applications*, Princeton Computer Science Notes, Princeton University Press, (1996).

[6] P. Martin-Löf, The definition of random sequences, *Inform. Control*, **9** (1966), 602–619.

[7] H. Sugita, Robust numerical integration and pairwise independent random variables, *Jour. Comput. Appl. Math.*, **139** (2002), 1–8.

[8] H. Sugita, Dynamic random Weyl sampling for drastic reduction of randomness in Monte Carlo integration, *Math. Comput. Simulation*, **62** (2003), 529–537.

[9] H. Sugita, Numerical integration of complicated functions and random sampling (in Japanese), "Sugaku", 56-1, Iwanami-shoten (2004), 1–17.(English translation, *SUGAKU EXPOSITIONS*, 19-2, AMS, December 2006, 153–169.)

[10] H. Sugita, Security of Pseudo-random Generator and Monte Carlo Method, *Monte Carlo Methods and Appl.*, **10-3**, VSP, (2004), 609–615.

[11] H. Sugita, *Monte Carlo method, Random number, and Pseudorandom number*, MSJ Memoirs vol.25 (2011), xiv+133 pp.

[12] H. Sugita, The Random Sampler, available at;
`http://www.math.sci.osaka-u.ac.jp/~sugita/mathematics.html`.

[13] H. Sugita and S. Takanobu, Random Weyl sampling for robust numerical integration of complicated functions, *Monte Carlo Methods and Appl.*, **6-1**, VSP, (1999), 27–48.

[14] A. Yao, Theory and applications of trapdoor functions, *Proceedings of the IEEE Foundations of Comput. Sci.*, (1982), 80–91.