Random Weyl sampling—A secure Monte Carlo integration method

Hiroshi Sugita

1 Department of Mathematics, Graduate school of Science, Osaka University
1-1, Machikaneyama, Toyonaka, Osaka, 560-0043, Japan

a) sugita@math.sci.osaka-u.ac.jp

Received October 27, 20XX; Revised December 29, 20XX; Published July 1, 20XX

Abstract: A proper mathematical formulation of the Monte Carlo method is presented. Based on it, a Monte Carlo integration method called Random Weyl sampling is proved to be secure, i.e., it is justifiable in rigorous mathematics.

Key Words: Monte Carlo integration, random number, secure pseudorandom generator, Random Weyl sampling, Dynamic random Weyl sampling

1. Introduction

How many times have we seen the following words?

Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.

These words that von Neumann addressed in a symposium([1]) and Knuth quoted in his famous book[2] might have misled many people to believe that any random sampling method by computer program should not be justifiable in rigorous mathematics. In fact, at least the Monte Carlo integration can be rigorously justified by using a proper pseudorandom generator.

The Monte Carlo method is easy to understand intuitively. But its mathematical formulation must not be intuitive, but must be as rigorous as that of algebra or geometry. A real number is usually regarded as a point in a straight line, but its rigorous definition is far from such an intuitive picture. Just like this, the mathematical formulation of the Monte Carlo method presented in § 2 could be far from the readers’ familiar pictures.

The mathematical formulation enables us to discuss things much more deeply than intuitive understanding can do. For example, to a question such as “Why is random number necessary for the Monte Carlo method?”, which is intuitively a trivial question, the mathematical formulation can give real significance and a very convincing answer. In particular, it enables us to describe clearly what property a pseudorandom generator should possess. It is “secureness” (§ 2.3). If a pseudorandom generator is secure, a Monte Carlo method using it can be justified with mathematical rigor. We call it a secure Monte Carlo method.

As a matter of fact, for the Monte Carlo integration, there exists a secure pseudorandom generator, and hence we have a secure Monte Carlo integration method. It is Random Weyl sampling (RWS for short) that realizes it in practice (§ 3, cf. [3, 4]). RWS is not only secure, but it shows a better...
convergence rate than the usual i.i.d.-sampling\(^1\) (§ 3.5 Theorem 7).

Most part of this article is taken from the author’s monograph[3]. The readers interested in this article are strongly recommended to read the book\(^2\).

2. Mathematical formulation

The Monte Carlo method is a numerical method to solve mathematical problems by computer-aided sampling of random variables. For each individual problem, we consider a random variable, and our aim is to get its generic value (Fig. 1)—a typical value or not an exceptional value—by sampling.

Fig. 1. The distribution and a generic value of a random variable

In this section, we explain several notions about the Monte Carlo method by considering the following concrete exercise.

Exercise 1 When we toss a coin 100 times, what is the probability \(p\) that Heads comes up at least 6 times in succession?

The strategy of the Monte Carlo method is as follows. Repeat independent trials of “100 coin tosses” many times, say \(10^6\) times, and let \(S_{10^6}\) be the relative frequency of the occurrences of “Heads comes up at least 6 times in succession” among all the trials. Then a generic value of \(S_{10^6}\) is close to the probability \(p\) by the law of large numbers.

2.1 Monte Carlo method

We need to set up a probability space to formulate Exercise 1. Since \(S_{10^6}\) is a function of \(100 \times 10^6 = 10^8\) coin tosses, the following probability space is suitable:

\[
\left(\{0,1\}^{10^8}, \mathcal{P}(\{0,1\}^{10^8}), P_{10^8}\right),
\]

where \(\{0,1\}^{10^8}\) is the set of all sequences of length \(10^8\) consisting of 0 (= Tails) and 1(= Heads), \(\mathcal{P}(\{0,1\}^{10^8})\) is the set of all subsets of \(\{0,1\}^{10^8}\), and \(P_{10^8}\) is the uniform probability measure on \(\{0,1\}^{10^8}\), i.e.,

\[
P_{10^8}(B) := \frac{\# B}{\# \{0,1\}^{10^8}} = \frac{\# B}{2^{10^8}}, \quad B \in \mathcal{P}(\{0,1\}^{10^8}) \quad (\text{i.e., } B \subset \{0,1\}^{10^8}).
\]

Now \(S_{10^6}\) is a random variable defined on this probability space, i.e., \(S_{10^6} : \{0,1\}^{10^8} \to \mathbb{R}\). To solve Exercise 1, we have to choose an \(\omega \in \{0,1\}^{10^8}\) and calculate the sample value \(S_{10^6}(\omega)\). According to the law of large numbers, if we set

\[
A := \left\{ \omega \in \{0,1\}^{10^8} \left| |S_{10^6}(\omega) - p| \leq \frac{1}{200} \right. \right\},
\]

\(^1\)The term i.i.d. stands for “independently identically distributed”.
\(^2\)For Japanese readers, [5] is recommended to read before [3].
the probability

\[ q := P_{10^8}(A) \]

is very close to 1. Indeed, by Chebyshev’s inequality (Eq. (6) below), we have

\[ q = P_{10^8}\left( |S_{10^8}(\omega) - p| \leq \frac{1}{200} \right) \geq \frac{99}{100}. \]

The Monte Carlo method is a kind of stochastic game, i.e., gambling. The player, say Alice, chooses an \( \omega \in \{0, 1\}^{10^8} \) of her own will. Her aim is to get a generic value of \( S_{10^8} \), which can be get if \( \omega \in A \). Thus the rule of the game is; if \( \omega \in A \), she wins, and if \( \omega \notin A \), she looses. Since the probability that she wins is \( q \geq 99/100 \), this game is very advantageous to her.

### 2.2 Random number

However, it is not easy for Alice to win the game. To choose an \( \omega \in \{0, 1\}^{10^8} \), she cannot help using a computer because of its huge amount of information; \( \omega \) is approximately a 12 MByte data. Obviously, a 12MByte data is too huge for anyone to input directly from a keyboard to a computer. So, she needs some program. But whatever program she may use, those \( \omega \)'s in \( \{0, 1\}^{10^8} \) she can choose of her own will are very limited. Let us assume that Alice can input at most 1,000 bit data directly from the key board. Then the number of \( \omega \)'s in \( \{0, 1\}^{10^8} \) she can choose is at most \( 2^{1000} \). (This is because the number of all the \( k \) bit data is \( 2^k \), and so the number of all data at most 1,000 bit is \( 2^1 + 2^2 + \cdots + 2^{1000} = 2^{1001} - 2 \).) Since the number of all the elements of \( \{0, 1\}^{10^8} \) is \( 2^{10^8} \), we know that how few those \( \omega \)'s in \( \{0, 1\}^{10^8} \) she can choose are. Although \( P_{10^8}(A) \) is very close to 1, since this probability is calculated under the assumption that Alice chooses every \( \omega \) in \( \{0, 1\}^{10^8} \) with equal probability, which is impossible for her, we cannot say that this game is advantageous to her in practice.

Let us look more closely at the situation. Even if she were able to input at most \( 10^8 - 10 \) bit data, the number of \( \omega \)'s in \( \{0, 1\}^{10^8} \) she could choose would be at most \( 2^{10^8-9} \), which is only 1/512 of all the \( \omega \)'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 chosen. If a long \( \{0, 1\} \)-sequence \( \omega \) requires an input which is almost as long as \( \omega \) itself to be chosen, it is called a random number ([6–10, 13]).

**Remark 2** It is quite difficult to get any knowledge about individual random numbers. But since random numbers account for nearly all long \( \{0, 1\} \)-sequences, it is a good idea to study characteristic properties that nearly all sequences share. Such properties have been studied very much in probability theory—properties described in various limit theorems, such as law of large numbers, central limit theorem, etc. Note that the set of random numbers and the event of probability close to 1 that a limit theorem specifies are not completely identical, but slightly different (Fig. 2).

To let the probability \( q = P_{10^8}(A) \) that Alice wins have a substantial meaning, she should choose \( \omega \in \{0, 1\}^{10^8} \) mainly among random numbers, which account for nearly all \( \{0, 1\}^{10^8} \)-sequences. This is the reason why it is said that random number is needed for the Monte Carlo method.

Although random numbers account for nearly all sequences, Alice cannot choose any one of them. As a result \( q = P_{10^8}(A) \) has no practical meaning. This is the most essential mathematical problem of the Monte Carlo method.

### 2.3 Pseudorandom number

Is random number absolutely necessary for the Monte Carlo method? No, it is not. Pseudorandom number may suffice.

A pseudorandom generator is a mapping which stretches short \( \{0, 1\} \)-sequences into long \( \{0, 1\} \)-sequences ([11]). To solve Exercise 1, suppose that Alice uses a pseudorandom generator \( g : \{0, 1\}^{238} \rightarrow \{0, 1\}^{10^8} \). She inputs a 238-bit data\(^3 \) \( \omega' \) from a keyboard to a computer. Then the computer calculates a \( 10^8 \)-bit data \( g(\omega) \) from her input \( \omega' \), and output a sample value \( S_{10^8}(g(\omega')) \). \( \omega' \) is called a seed, and

\(^3\)The origin of the number 238 will be clear in § 3.3.
The whole \{0,1\} -sequences \ The set of random numbers
The event of probability close to 1 that a limit theorem specifies

Fig. 2. Random number and limit theorem

\( g(\omega') \) the corresponding pseudorandom number. Note that \( g(\omega') \in \{0, 1\}^{10^8} \) is not a random number, because it is generated from the much shorter seed \( \omega' \in \{0, 1\}^{2^{38}} \).

Alice is now betting on whether \( g(\omega') \in A \) or not. This time, the probability that she wins is

\[
q' := P_{238} \left( |S_{10^6}(g(\omega')) - p| \leq \frac{1}{200} \right) = \frac{\#\{\omega' \in \{0, 1\}^{2^{38}} | g(\omega') \in A\}}{2^{38}},
\]

which has a practical meaning, since she can actually choose any \( \omega' \in \{0, 1\}^{2^{38}} \) of her own will. Of course, this probability \( q' \) depends on \( g \), but if there exists a pseudorandom generator \( g \) such that \( q' \) remains very close to 1, Alice can actually win the game with high probability. Namely, using \( g \) does not make the risk that she loses bigger. Such a \( g \) is said to be secure for the event \( A \) or the random variable \( S_{10^6} \) (cf. [11]). Thus Exercise 1 will be solved by finding a secure pseudorandom generator.

In general, a Monte Carlo method using a secure pseudorandom generator is called a secure Monte Carlo method. For a general random variable, we do not know if there is a pseudorandom generator which is secure for it. However, if we restrict the use of pseudorandom generator to the Monte Carlo integration, i.e., if the random variable in question is a sample mean of i.i.d. random variables, there exists a pseudorandom generator which is secure for it. In other words, we already have a secure Monte Carlo integration method in practice.

3. Random Weyl sampling

In this section, we introduce a secure Monte Carlo integration method—Random Weyl sampling (RWS)—developed by [3, 4]. The point is that the sample variables that RWS generates are not mutually independent, but pairwise independent.

3.1 Monte Carlo integration

Apart from Exercise 1, we consider a general setup. Let \( m \) be a positive integer, and let

\[
(\{0, 1\}^m, \mathcal{B}(\{0, 1\}^m), P_m)
\]

be the probability space for \( m \) coin tosses. Suppose now we want to calculate the mean \( E[X] \) of a random variable \( X \) defined on this probability space;

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

If \( m \) is a little bit large, say \( m = 100 \), the sum of the right hand side cannot be calculated in practice. Then we utilize the law of large numbers to estimate the mean; we repeat \( N \) independent trials of \( \sim m \)
coin tosses". For each trial, we evaluate the sample value of $X$, and compute the arithmetical mean $S_N$ of all the $N$ samples. The rigorous formulation is as follows; $S_N$ is a random variable defined on the probability space

$$
\left(\{0,1\}^N, \mathcal{F}(\{0,1\}^N), P_{Nm}\right)
$$

of $Nm$ coin tosses, by

$$
S_N(\omega) := \frac{1}{N} \sum_{n=1}^{N} X(\xi_n),
$$

where

$$
\omega = (\xi_1, \ldots, \xi_N) \in \{0,1\}^N, \quad \xi_n \in \{0,1\}^m, \quad n = 1, \ldots, N.
$$

Then the law of large numbers (Chebyshev’s inequality) shows that

$$
P_{Nm}(|S_N(\omega) - \mathbb{E}[X]| \leq \varepsilon) \geq 1 - \frac{\mathbb{V}[X]}{N\varepsilon^2}, \quad \varepsilon > 0,
$$

where

$$
\mathbb{V}[X] := \mathbb{E}[(X - \mathbb{E}[X])^2]
$$

is the variance of $X$.

Since the right hand side of Eq. (1) is a sum of i.i.d. random variables, we call this sampling method i.i.d.-sampling.

### 3.2 RWS: definition and theorem

Let

$$
D_m := \{i/2^m | i = 0, 1, 2, \ldots, 2^m - 1\} \subset [0,1).
$$

Accordingly, we define $[\bullet]_m$ as

$$
[x]_m := \lfloor 2^m(x - \lfloor x \rfloor) \rfloor / 2^m \in D_m, \quad x \geq 0,
$$

where $\lfloor x \rfloor$ denotes the largest integer not greater than $x$.

**Definition 3** Let $j$ be a positive integer. For each

$$
\omega' = (x, \alpha) \in D_{m+j} \times D_{m+j}
$$

we define $\{Z_n(\omega')\}_{n=1}^{2^{j+1}}$ by

$$
Z_n(\omega') := [x + n\alpha]_m \in D_m, \quad 1 \leq n \leq 2^{j+1}.
$$

By the binary expansion mapping $\psi_m : \{0,1\}^m \rightarrow D_m$ defined by

$$
\psi_m(\xi) := \sum_{i=1}^{m} \frac{\xi_i}{2^i} \in D_m, \quad \xi = (\xi_1, \ldots, \xi_m) \in \{0,1\}^m,
$$

we will identify $\{0,1\}^m$ with $D_m$, which will be written as $\{0,1\}^m \cong D_m$. Accordingly, let $P_m$ also mean the uniform probability measure on $D_m$.

Then noting that $\omega' \in D_{m+j} \times D_{m+j} \cong \{0,1\}^{m+j} \times \{0,1\}^{m+j}$ and that $\{0,1\}^{m+j} \times \{0,1\}^{m+j}$ can be identified with $\{0,1\}^{2m+2j}$ in an obvious way, we will think that $\omega' \in \{0,1\}^{2m+2j}$. Accordingly, the sequence $\{Z_n(\omega')\}_{n=1}^{2^{j+1}}$ is now regarded as a sequence of random variables defined on the probability space $\left(\{0,1\}^{2m+2j}, \mathcal{F}(\{0,1\}^{2m+2j}), P_{2m+2j}\right)$.

Under these interpretations, we have the following theorem.

**Theorem 4** The sequence of random variables $\{Z_n(\omega')\}_{n=1}^{2^{j+1}}$ is pairwise independent, i.e., if $1 \leq n < n' \leq 2^{j+1}$, then $Z_n(\omega')$ and $Z_{n'}(\omega')$ are independent. Furthermore, each $Z_n(\omega')$ is distributed uniformly in $D_m$.  

1105
Theorem 4 has been proved by [3, 4, 12]. Here we present another new and even easier proof. First we quote an elementary algebraic lemma without proof.

**Lemma 5** ([14], Theorem 25) *Let a and b be coprime integers. Then there exist integers x and y that satisfy an indeterminate equation ax + by = 1.*

**Proof of Theorem 4.** For any pair \(1 \leq n_1 < n_2 \leq 2^{j+1}\), we will show that

\[
P_{2m+2j}(Z_{n_1}(\omega') = u, Z_{n_2}(\omega') = v) = 2^{-2m}, \quad u, v \in D_m \cong \{0, 1\}^m.
\]

To this end, we define

\[
A(u, v) := \{(x, \alpha) \in D_{m+j} \times D_{m+j} \mid [x + n_1 \alpha]_m, [x + n_2 \alpha]_m) = (u, v)\}, \quad u, v \in D_m,
\]

and we will show that

\[
\frac{\#A(u, v)}{2^{2m+2j}} = 2^{-m}, \quad u, v \in D_m,
\]

or equivalently, we will show that

\[
\#A(u, v) = \#A(0, 0), \quad u, v \in D_m. \quad (3)
\]

First, let us show that there exists a translation \(T_1 : D_{m+j}^2 \times D_{m+j}^2 \to D_{m+j}^2 \times D_{m+j}^2\)

of the type

\[
T_1 : (x, \alpha) \mapsto ([x + \Delta x]_m, [\alpha + \Delta \alpha]_m)
\]

such that

\[
T_1 A(u, v) = A(u + 2^{-m}, v), \quad u, v \in D_m.
\]

The system of equations that the increments \((\Delta x, \Delta \alpha)\) should satisfy is

\[
[x + \Delta x + n_1(\alpha + \Delta \alpha)]_m = [x + n_1 \alpha]_m + 2^{-m},
\]

\[
[x + \Delta x + n_2(\alpha + \Delta \alpha)]_m = [x + n_2 \alpha]_m,
\]

which follows from

\[
\Delta x + n_1 \Delta \alpha \equiv 2^{-m}, \quad (4)
\]

\[
\Delta x + n_2 \Delta \alpha \equiv 0, \quad (5)
\]

where "≡" means that the difference of the both hand sides is an integer. Subtracting Eq. (4) from Eq. (5), we have

\[
(n_2 - n_1)\Delta \alpha \equiv -2^{-m}.
\]

We can write \(n_2 - n_1 = 2^l s\), where \(s\) is an odd integer and \(0 \leq l \leq j\). Then the above equation is reduced to

\[
2^l s \Delta \alpha = t - 2^{-m}, \quad \text{for some } t \in \{1, \ldots, n_2 - n_1\}.
\]

Multiplied this by \(2^m\), we get an indeterminate equation with unknown integers \(2^{m+l} \Delta \alpha\) and \(t\)

\[
s \cdot (2^{m+l} \Delta \alpha) - 2^m \cdot t = -1.
\]

Since \(2^m\) and \(s\) are coprime, Lemma 5 implies that this equation has a pair of integer solutions \((2^{m+l} \Delta \alpha, t)\). This means that the system of Eqs. (4)(5) has a solution

\[
(\Delta x, \Delta \alpha) \in D_{m+l} \times D_{m+l} \subset D_{m+j} \times D_{m+j}.
\]

Thus, the translation \(T_1\) definitely exists.
Similarly, we can prove the existence of a translation \( T_2 : D_{m+j} \times D_{m+j} \rightarrow D_{m+j} \times D_{m+j} \) which maps \( A(u, v) \) onto \( A(u, v + 2^{-m}) \). Since \((0, 0) \in A(0, 0)\), we see \( A(0, 0) \neq \emptyset \). Using the translations \( T_1 \) and \( T_2 \), we have
\[
A(u, v) = (T_1)^2 u (T_2)^2 v A(0, 0), \quad u, v \in D_m.
\]
Since the translations are bijective, we see all \( A(u, v) \) have the same number (\( = 2^{j-l} \)) of elements, i.e., Eq. (3) holds. \( \square \)

As a consequence of Theorem 4, we have the following theorem.

**Theorem 6** Let \( 2 \leq N \leq 2^{j+1} \). Define a pseudorandom generator
\[
g : D_{m+j} \times D_{m+j} \cong \{0, 1\}^{2m+2j} \rightarrow (D_m)^N \cong \{0, 1\}^{Nm}
\]
by
\[
g(\omega') := (Z_1(\omega'), \ldots, Z_N(\omega')) , \quad \omega' \in D_{m+j} \times D_{m+j} \cong \{0, 1\}^{2m+2j}.
\]
Then the following two equalities hold.
\[
\mathbb{E}[S_N(g(\omega'))] = \mathbb{E}[S_N(\omega)] = \mathbb{E}[X],
\]
\[
\mathbb{V}[S_N(g(\omega'))] = \mathbb{V}[S_N(\omega)] = \frac{\mathbb{V}[X]}{N}.
\]
Consequently, Chebyshev’s inequality gives us the same estimation as Eq. (2):
\[
P_{2m+2j}(|S_N(g(\omega')) - \mathbb{E}[X]| \leq \varepsilon) \geq 1 - \frac{\mathbb{V}[X]}{N \varepsilon^2}, \quad \varepsilon > 0.
\]
This means that \( g \) is secure for \( S_N \). The secure Monte Carlo integration method using this pseudorandom generator \( g \) is called Random Weyl sampling (RWS for short).

**Proof.** By definition, we have
\[
S_N(g(\omega')) = \frac{1}{N} \sum_{n=1}^{N} X(Z_n(\omega')).
\]
That each \( Z_n(\omega') \) is distributed uniformly in \( D_m \) means \( X(Z_n(\omega')) \) and \( X(\xi) \) are identically distributed, and hence
\[
\mathbb{E}[S_N(g(\omega'))] = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}[X(Z_n(\omega'))] = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}[X] = \mathbb{E}[X].
\]
As for the variance, since \( X(Z_n(\omega')) \) and \( X(Z_{n'}(\omega')) \) are independent \((n < n')\), we have
\[
\mathbb{V}[S_N(g(\omega'))] = \frac{1}{N^2} \mathbb{E} \left[ \left( \sum_{n=1}^{N} (X(Z_n(\omega')) - \mathbb{E}[X]) \right)^2 \right]
\]
\[
= \frac{1}{N^2} \sum_{n=1}^{N} \sum_{n'=1}^{N} \mathbb{E} [(X(Z_n(\omega')) - \mathbb{E}[X]) (X(Z_{n'}(\omega')) - \mathbb{E}[X])]
\]
\[
= \frac{1}{N^2} \sum_{n=1}^{N} \mathbb{E} [(X(Z_n(\omega')) - \mathbb{E}[X])^2] + \frac{2}{N^2} \sum_{1 \leq n < n' \leq N} \mathbb{E} [(X(Z_n(\omega')) - \mathbb{E}[X]) \mathbb{E} [(X(Z_{n'}(\omega')) - \mathbb{E}[X])]
\]
\[
= \frac{1}{N^2} \sum_{n=1}^{N} \mathbb{E} [(X(Z_n(\omega')) - \mathbb{E}[X])^2]
\]
\[
= \frac{1}{N^2} \sum_{n=1}^{N} \mathbb{V}[X] = \frac{\mathbb{V}[X]}{N}.
\] \( \square \)
### 3.3 Solution to Exercise 1

Let us solve Exercise 1 by RWS. Let \( m = 100 \) and \( N = 10^6 \). Our target integrand is

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

This means \( X = 1 \) if there are 6 successive 1’s in \((\xi_1, \ldots, \xi_{100})\), and that \( X = 0 \) otherwise. Note that

\[
E[X] = P_{100}(X(\xi) = 1) = p, \quad V[X] = p(1 - p) \leq \frac{1}{4}.
\]

Next we define \( S_{10^6} : \{0, 1\}^{10^6} \rightarrow \mathbb{R} \) by

\[
S_{10^6}(\omega) := \frac{1}{10^6} \sum_{n=1}^{10^6} X(\xi_n),
\]

where

\[
\omega = (\xi_1, \ldots, \xi_{10^6}) \in \{0, 1\}^{10^6}, \quad \xi_n \in \{0, 1\}^{100}, \quad n = 1, \ldots, 10^6.
\]

Chebyshev’s inequality Eq. (2) for \( S_{10^6} \) is

\[
P_{10^6}\left(|S_{10^6}(\omega) - p| \leq \frac{1}{200}\right) \geq \frac{99}{100}. \tag{6}
\]

Let us take \( j := 19 \) so that \( 10^6 = N < 2^{j+1} = 2^{20} \). Hence the pseudorandom generator of RWS is now

\[g : D_{119} \times D_{119} \cong \{0, 1\}^{238} \rightarrow (D_{100})^{10^6} \cong \{0, 1\}^{10^6}.
\]

According to Theorem 6, we have

\[
P_{238}\left(|S_{10^6}(g(\omega')) - p| \leq \frac{1}{200}\right) \geq \frac{99}{100}.
\]

Since Alice can easily choose any seed \( \omega' \in \{0, 1\}^{238} \), she no longer needs a long random number.

![Diagram](image)

**Fig. 3.** The function of the pseudorandom generator \( g : \{0, 1\}^{238} \rightarrow \{0, 1\}^{10^6} \).

Here is a concrete example. Instead of Alice, the author chose the following seed \( \omega' = (x, \alpha) \in D_{119} \times D_{119} \cong \{0, 1\}^{238} \) written in binary expansion;
Then the computer calculated as $S_{106}(g(\omega')) = 0.546177$, which is expected to be an approximated value of the probability $p$.

The above numerical result was obtained by the following C program.

```c
/*====================================================*/
/* file name: exercise_1.c */
/*====================================================*/
#include <stdio.h>
#define SAMPLE_NUM 1000000
#define M 100
#define M_PLUS_J 119
/* seed */
char xch[] =
    "1110110101" "1011101101" "0100000011" "0110101001"
    "0101000100" "0101111101" "1010000000" "1010100011"
    "0100011001" "1101111101" "1101010011" "111100100";
char ach[] =
    "1100000111" "0111000100" "0001101011" "1001000001"
    "0010001000" "1010101101" "1110101110" "0010010011"
    "1000000011" "0101000110" "0101110010" "010111111";
int x[M_PLUS_J], a[M_PLUS_J];

void longadd(void) /* x = x + a (long digit addition) */
{
    int i, s, carry = 0;
    for ( i = M_PLUS_J-1; i >= 0; i-- ){
        s = x[i] + a[i] + carry;
        if ( s >= 2 ) {carry = 1; s = s - 2; } else carry = 0;
        x[i] = s;
    }
}

int maxLength(void) /* count the longest run of 1's */
{
    int len = 0, count = 0, i;
    for ( i = 0; i <= M-1; i++ ){
        if ( x[i] == 0 )
            { if ( len < count ) len = count; count = 0;}
        else count++;
    }
    return len;
}
```

1109
```c
int main()
{
    int n, s = 0;
    for( n = 0; n <= M_PLUS_J-1; n++ ){
        if( xch[n] == '1' ) x[n] = 1; else x[n] = 0;
        if( ach[n] == '1' ) a[n] = 1; else a[n] = 0;
    }
    for ( n = 1; n <= SAMPLE_NUM; n++ ){
        longadd();
        if ( maxLength() >= 6 ) s++;
    }
    printf ( "s=%6d, p=%7.6f
", s, (double)s/(double)SAMPLE_NUM);
    return 0;
}
/*================ End of exercise_1.c ===============*/
```

### 3.4 From viewpoint of mathematical statistics

We have formulated the Monte Carlo method as gambling and we have considered that the seed \(\omega' \in \{0,1\}^{238}\) of a pseudorandom generator \(g\) is chosen by Alice of her own will (§ 2.1). But from the viewpoint of mathematical statistics, this is not a good formulation, because sampling should be done unintentionally in order to guarantee the objectivity of the result. Indeed, in the case of RWS, Alice can choose a bad seed on purpose, say \(\omega = \langle 0, \ldots, 0 \rangle\), 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 \(\omega' \in \{0,1\}^{238}\), we toss a coin 238 times, record 1 if Heads comes up and 0 if Tails does at each coin toss, define \(\omega'\) as the recorded \(\{0,1\}\)-sequence, and finally compute \(S(g(\omega'))\), which completes an objective sampling. As a matter of fact, RWS in § 3.3 was performed in this way.

This method cannot be used to choose a very long \(\omega \in \{0,1\}^{10^8}\). The point is that the pseudorandom generator \(g : \{0,1\}^{238} \to \{0,1\}^{10^8}\) makes the input shorter so that this method may become executable.

### 3.5 Convergence theorem

The law of large numbers holds for pairwise independent random variables (Theorem 6), but the central limit theorem does not necessarily hold for them ([15]). In the case of (the continuous version of) RWS, we have the following theorem.

**Theorem 7 ([4, 15])** Let \(f : \mathbb{R} \to \mathbb{R}\) be any periodic Borel measurable function of period 1 satisfying \(\int_0^1 |f(x)|^2 dx < \infty\). Then for any \(1 \leq p < 2\), it holds that

\[
\lim_{N \to \infty} \int_0^1 \int_0^1 \left| \frac{1}{\sqrt{N}} \sum_{n=1}^N \left( f(x + n\alpha) - \int_0^1 f(y) dy \right) \right|^p \, d\alpha \, dx = 0.
\]

Consequently, for any \(\varepsilon > 0\), \(\lambda^2\) being the 2-dimensional Lebesgue measure, it holds that

\[
\lambda^2 \left( \left\{ (x, \alpha) \in [0,1) \times [0,1) \left| \left| \frac{1}{\sqrt{N}} \sum_{n=1}^N \left( f(x + n\alpha) - \int_0^1 f(y) dy \right) \right| > \varepsilon \right\} \right) \to 0, \quad N \to \infty.
\]

i.e., the limit distribution of the central limit theorem scaling of the sample mean degenerates.

By virtue of Theorem 7, the sample mean calculated by RWS converges to the mean faster than the usual i.i.d.-sampling.
Example 8 To see the effect of Theorem 7, we computed the distribution of $S_{10\varphi}(g(\omega'))$ by a Monte Carlo method. Using the pseudorandom generator by means of Weyl transformation([4, 17]), we generated 50,000 random seeds $\omega' = (x, \alpha) \in D_{119} \times D_{119}$, and investigated the frequency distribution of $S_{10\varphi}(g(\omega'))$.

![Figure 4. The frequency distribution of 50,000 samples of $S_{10\varphi}(g(\omega'))$](image)

Figure 4 shows the frequency distribution of 50,000 samples of $S_{10\varphi}(g(\omega'))$. In this figure, the thick curve shows the normal density function of $N(0.546095, 0.000262653^2)$. Comparing with this, the distribution of $S_{10\varphi}(g(\omega'))$ is more concentrated around the mean and has thicker tails. The thin curve is the normal density function of $N(0.546095, 0.000497871^2)$, which exactly approximates the distribution of $S_{10\varphi}(\omega)$. Obviously, RWS is much preferable to i.i.d.-sampling.

4. Concluding remarks

RWS is a secure Monte Carlo integration method. But it is only applicable to random variables that are functions of $m$ coin tosses for some fixed positive integer $m$, e.g., in Exercise 1, $m = 100$. In fact, there are random variables that are functions of a finite number of coin tosses with probability 1, but the number cannot be fixed in advance. For example, consider the following exercise.

Exercise 9 What is the expected number of coin tosses before 5 Heads are obtained?

To solve such a problem, the author developed the Dynamic Random Weyl sampling (DRWS for short) as an extended version of RWS. The sequence of random variables that DRWS generates is pairwise independent but not mutually independent. As far as the author knows, DRWS is the most useful Monte Carlo integration method. For details, see [4, 16].

References

[1] J. von Neumann, “Various techniques used in connection with random digits,” U.S. Natl. Bur. Stand. Appl. Math. Ser., vol. 12, pp. 36–38, 1951.
[2] D.E. Knuth, The Art of Computer Programming, 3rd ed., Volume 2, Chapter 3, Random numbers, Addison-Wesley, 1997.
[3] H. Sugita, “Robust numerical integration and pairwise independent random variables,” Jour. Comput. Appl. Math., vol. 139, pp. 1–8, 2002.
[4] H. Sugita, Monte Carlo Method, Random Number, and Pseudorandom Number, MSJ Memoirs vol. 25, World Scientific, 2011.
[5] H. Sugita, *Probability and Random number (Kakuritsu to ransū in Japanese)*, Sūgaku shobō, 2015.

[6] G.J. Chaitin, “Algorithmic information theory,” *IBM J. Res. Develop.*, vol. 21, pp. 350–359, 1977.

[7] A.N. Kolmogorov, “Three approaches to the definition of the concept quantity of information,” (Russian) *Problemy Peredači Informacii* vol. 1, no. 1, pp. 3–11, 1965.

[8] A.N. Kolmogorov, “Logical basis for information theory and probability theory,” *IEEE Trans. on Inform. Theo.*, vol. IT-14, no. 5, pp. 662–664, September 1968.

[9] P. Martin-Löf, “The definition of random sequences,” *Inform. Control* vol. 9, pp. 602–619, 1966.

[10] R. Solomonoff, “A formal theory of inductive inference. Part 1,” *Inform. Control* vol. 7, pp. 1–22, 1964.

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

[12] M. Dietzfelbinger, “Universal hashing and k-wise independent random variables via integer arithmetic without primes,” *Lecture Notes in Computer Science* vol 1046, pp. 569–580, 1996.

[13] R.G. Downey and D.R. Hirschfeld, *Algorithmic Randomness and Complexity*, Springer, 2010.

[14] G.H. Hardy and E.M. Wright, *An introduction to the theory of numbers*, Fifth edition, Oxford University Press, 1979.

[15] S. Janson, “Some pairwise independent sequences for which the central limit theorem fails,” *Stochastics* vol. 23, no. 4, pp. 439–448, 1988.

[16] H. Sugita, “Dynamic random Weyl sampling for drastic reduction of randomness in Monte Carlo integration,” *Math. Comput. Simulation*, vol. 62, pp. 529–537, 2003.

[17] H. Sugita, “Pseudo-random number generator by means of irrational rotation,” *Monte Carlo Methods and Applications*, vol.1, no.1, pp.35–57, VSP, 1995.