# THE MONTE CARLO ALGORITHM WITH A PSEUDO–RANDOM GENERATOR

J. F. Traub

Department of Computer Science, Columbia University

H. Woźniakowski

Department of Computer Science, Columbia University and Institute of Informatics, University of Warsaw

CUCS-030-92

December, 1989

Abstract. We analyze the Monte Carlo algorithm for the approximation of multivariate integrals when a pseudo-random generator is used. We establish lower and upper bounds on the error of such algorithms. We prove that as long as a pseudo-random generator is capable of producing only finitely many points, the Monte Carlo algorithm with such a pseudo-random generator fails for $L_2$ or continuous functions. It also fails for Lipschitz functions if the number of points does not depend on the number of variables. This is the case if a linear congruential generator is used with one initial seed. On the other hand, if a linear congruential generator of period $m$ is used for each component with independent uniformly distributed initial seeds, then the Monte Carlo algorithm with such a pseudo-random generator using $n$ function values behaves as for the uniform distribution and its expected error is roughly $n^{-1/2}$ as long as the number $n$ of function values is less than $m^2$.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

## 1. Introduction

Randomization is being widely used or proposed to solve both continuous and discrete problems. Examples include multivariate integration, algebraic eigenvalues, primality testing, byzantine agreement, and verification. When randomized algorithms are implemented on a computer, pseudo-random numbers must be used. In this paper we investigate whether the good properties of the Monte Carlo algorithm for multivariate integration hold if pseudo-random numbers are used. We suggest that such an analysis should be performed whenever randomization is used.

One can identify two types of work regarding pseudo-random generators. In the first, they are studied in isolation from a particular problem. Excellent surveys are given in Knuth [81, Chapter 3] and Niederreiter [78, 88, 89]. More recently, polynomial-time unpredictability of pseudo-random generators has been studied by Yao [82], Blum and Micali [84] and others.

In the second, the relation between pseudo-random generators and randomized algorithms is studied for a specific problem. Examples are provided by Bach [87] who studied finding square roots modulo a prime number and primality testing. Karloff and Raghavan [88] studied sorting, selection and oblivious routing in networks. They showed that certain randomized algorithms work well with a linear congruential generator and a random seed.

In this paper we discuss approximate integration of functions of $d$ variables. It is known that for $r$ times differentiable functions we have to compute $\Theta(\varepsilon^{-d/r})$ function values at deterministically chosen points to guarantee an $\varepsilon$-approximation to the integral in the worst case setting, see Section 2. If $d$ is large relative to $r$ the problem is intractable since even the fastest computers cannot compute so many function values.

Many other problems have been proven to be intractable. Sometimes randomization may be used to break intractability; a general discussion of randomization may be found in Traub, Wasilkowski and Woźniakowski [88, Chapter 11].

For multivariate integration, randomization is used by computing function values at randomized points. In this paper we shall study the *Monte Carlo algorithm with a distribution* $\rho$. That is, we approximate the integral $\int_{[0,1]^d} f(t)\,dt$ by the arithmetic mean $n^{-1}\sum_{i=1}^{n} f(t_i)$, where the points $t_i$ are random variables drawn with respect to $\rho$. As is discussed below, we can treat pseudo-random generators as random number generators with respect to some distribution, and alternatively we will also write the *Monte Carlo algorithm with a pseudo-random generator.*

When the distribution $\rho$ is uniform, i.e., the points $t_i$ are independent and uniformly distributed random variables over the unit cube, we omit mentioning $\rho$ and refer simply to the *Monte Carlo algorithm.* Thus when we write *Monte Carlo algorithm* we mean the classical algorithm with independent and uniformly distributed points $t_i$. It is known that the Monte Carlo algorithm enjoys the following good properties:

(1) convergence rate $n^{-1/2}$ independent of dimension $d$,
(2) the convergence rate holds even for functions from $L_2$.

In computational practice, randomized points are obtained by using a pseudo-random generator, such as a linear congruential generator, from which the evaluation points are computed. Obviously, the use of a pseudo-random generator can at best approximate independent and uniformly distributed random variables.

The problem addressed in this paper is whether the good properties of the Monte Carlo algorithm can be preserved if a pseudo-random generator is used. A pseudo-random generator may be understood as a deterministic mechanism to generate points. This deterministic mechanism may depend on some parameters. For instance, a linear congruential generator depends on an initial seed. If these parameters are randomly selected, the corresponding pseudo-random generator can be formally treated as a random generator with respect to some distribution. The assumption of random selection of parameters may or may not be realistic. Observe that if randomness is not allowed the worst case results imply intractability of multivariate integration. That is why we assume in this paper that a pseudo-random generator depends on some randomly selected parameters and we treat pseudo-random generators as random generators with respect to some distribution.

We now informally summarize the conclusions of this paper. We prove that the Monte Carlo algorithm with an arbitrary pseudo-random generator which is capable of producing only finitely many points fails for some continuous functions, see Theorem 3.1 This negative result means that the use of such pseudo-random generators requires more smoothness than the Monte Carlo algorithm.

On the other hand we would like to assume as little extra smoothness as possible. We compromise by considering the class of Lipschitz functions with uniformly bounded Lipschitz constant. Without loss of generality we take the constant as unity.

For this class we obtain lower and upper bounds on the error of the Monte Carlo algorithm with a pseudo-random generator.

We first discuss lower bounds. In Section 3, we show that the error of the Monte Carlo algorithm with an arbitrary pseudo-random generator is at least about $(k^*)^{-1/d}$, see (3.8), where $k^*$ denotes the total number of points which the pseudo-random generator is capable of producing.

In particular, assume that a linear congruential generator of period $m$ is used with one randomly selected initial seed and that the points at which the function is evaluated are defined from $d$ successive pseudo-random numbers. Then $k^* = m$ and $k^*$ is independent of $d$, see Example 3.1. Note that even for large $m$ and modest $d$, $(k^*)^{-1/d}$ is not small. In this case, the algorithm again fails.

To ensure small error of the Monte Carlo algorithm with a pseudo-random generator for the class of Lipschitz functions, we must guarantee that $(k^*)^{-1/d}$ is small even for large $d$. This can be achieved if a linear congruential generator of period $m$ is used $d$ times for each component, with *independent* initial seeds. We then have $k^* = m^d$ and $(k^*)^{-1/d} = m^{-1}$, see Example 3.2 and also Example 3.3. Usually, $m$ is large, say $m = 2^{30}$, and the lower bound is quite acceptable.

We now turn to upper bounds. In Section 4, we establish upper bounds on the error of the Monte Carlo algorithm with points which are independent random variables produced by arbitrary pseudo-random generators. In particular, the upper bound for a linear congruential generator of period $m$ used $d$ times for each component independently and with uniform distribution is roughly $n^{-1/2}$ as long as $n \le m^2$, see Theorems 4.1 and 4.2. In this case, behavior is essentially the same as with uniform distribution.

In Section 5 we mention a few open problems. We suggest carrying out similar analyses for problems such as weighted multivariate integration and algebraic eigenvalues. It would be also of interest to analyze different pseudo-random generators, in particular, generators which, at least theoretically, are capable of producing infinitely many points.

## 2. Determinism and Randomization for Multivariate Integration

Let $T = [0,1]^d$ be the $d$-dimensional unit cube. We wish to approximate the multivariate integral

$$I(f) = \int_T f(t)\,dt \tag{2.1}$$

for any function $f : T \to \mathbb{R}$ which belongs to a class $F$, where $F$ is a subset of $L_2(T)$.

We first consider deterministic algorithms $\phi_n$ which use $n$ function values at deterministically chosen points to approximate $I(f)$. More precisely, $\phi_n$ is a mapping (not necessarily linear) of the form $\phi_n(f(t_1), f(t_2), \ldots, f(t_n))$, where the deterministic points $t_1, t_2, \ldots, t_n$ belong to $T$ and may be chosen adaptively, i.e., $t_i$ may depend on the already computed $f(t_1), f(t_2), \ldots, f(t_{i-1})$. Let

$$e^{\det}(\phi_n, F) = \sup_{f \in F} |I(f) - \phi_n(f(t_1), \ldots, f(t_n))|$$

be the deterministic worst case error of the algorithm $\phi_n$. Let

$$e_n^{\det}(F) = \inf_{\phi_n} e^{\det}(\phi_n, F) \tag{2.2}$$

be the $n$th deterministic minimal error which can be achieved by using $n$ function values.

The quantity $e_n^{\det}(F)$ has been extensively studied for many different classes $F$ of multivariate functions, see e.g., Novak [88] for a recent survey. For instance, if $F = BL_2(T)$ is the unit ball of the space of $L_2$-integrable functions, or $F = C_d$ is the unit ball of the class of continuous functions with sup norm, then

$$e_n^{\det}(BL_2(T)) = e_n^{\det}(C_d) = 1, \quad \forall n. \tag{2.3}$$

This easily follows by noting that if $f$ belongs to $C_d$ and takes on, say, zero values at $n$ points then the best bounds on its integral is the interval $(-1, +1)$ and the best approximation is zero, i.e., the midpoint point of this interval. This implies that the error of any algorithm is at least 1, and since the zero algorithm, $\phi_n \equiv 0$, has error 1, we have $1 = e_n^{\det}(C_d) \leq e_n^{\det}(BL_2(T)) \leq 1$, as claimed. Thus, it is impossible to approximate $I(f)$ with error less than one no matter how many function values are used.

This negative result can be interpreted as stating that continuity of $f$ is not enough to compute an approximation in the deterministic worst case setting. Therefore, it is natural to study classes of smoother functions. Consider

$$C_d^{r,\alpha} = \{ f : T \to \mathbb{R} \ : \ |D^{(j)}f(x) - D^{(j)}f(y)| \leq \max_{1\leq i\leq d} |x_i - y_i|^{\alpha}, \quad \forall |j| \leq r \} \tag{2.4}$$

as the class of functions all of whose partial derivatives up to order $r$, $r$ is a nonnegative integer, satisfy Hölder condition with $\alpha \in (0, 1]$.

Bakhvalov [59] proved that for $F = C_d^{r,\alpha}$ we have

$$e_n^{\det}(C_d^{r,\alpha}) = \Theta(n^{-(r+\alpha)/d}), \quad \text{as } n \to +\infty, \tag{2.5}$$

see also Novak [88, p.34]. Thus the $n$th minimal error goes to zero. However, if $d$ is large relative to $r + \alpha$ then the rate of convergence is very poor. To see this more clearly, let $n = n^{\det}(\varepsilon)$ be the smallest integer for which $e_n^{\det}(C_d^{r,\alpha}) \leq \varepsilon$. Then

$$n^{\det}(\varepsilon) = \Theta\left( \left( \varepsilon^{-1/(r+\alpha)} \right)^d \right). \tag{2.6}$$

Thus $n^{\det}(\cdot)$ is an exponential function of $d$, and for $d$ large relative to $r + \alpha$, $n^{\det}(\varepsilon)$ is huge even for moderate $\varepsilon$. In this case, multivariate integration is intractable since it is impossible to compute so many function values. One may say that the "curse of dimensionality" makes the problem intractable.

To break intractability of multivariate integration one can evaluate $f$ at randomized rather than deterministic points. In this paper we shall study the *Monte Carlo algorithm with distribution* $\rho$. That is we approximate $\int_T f(t)\,dt$ by

$$\phi_n^{\mathrm{MC}}(f;t_1,\dots,t_n)=\frac{1}{n}\sum_{i=1}^n f(t_i), \tag{2.7}$$

where the points $t_1,\dots,t_n$ are random variables drawn with respect to $\rho$.

When the distribution $\rho$ is uniform, i.e., the points $t_i$ are independent and uniformly distributed random variables over $T$, we omit mentioning $\rho$ and refer simply to the *Monte Carlo algorithm.* There is a huge literature on randomization with applications to many diverse problems. See books and surveys by Hammersley and Handscomb [64], Granovskii and Ermakov [77], Davis and Rabinowitz [84], Sobol [85], Kalos and Whitlock [86], Niederreiter [78, 88, 89], and Novak [88].

The quality of the Monte Carlo algorithm is defined by the expected error with respect to the points $t_i$ for a worst case function $f$ from the class $F$. That is,

$$e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}},F,\lambda)=\sup_{f\in F}\left(\int_{T^n}\left(I(f)-\frac{1}{n}\sum_{i=1}^n f(t_i)\right)^2 dt_1\dots dt_n\right)^{1/2}, \tag{2.8}$$

where $\lambda$ stands for the Lebesgue measure.

The basic property of the Monte Carlo algorithm is that the expected error $E_n(f,\lambda)$ for a function $f$ is

$$E_n(f,\lambda)=\left(\int_{T^n}\left(I(f)-\frac{1}{n}\sum_{i=1}^n f(t_i)\right)^2 dt_1\dots dt_n\right)^{1/2}=\frac{1}{\sqrt{n}}\left(\int_T (f(t)-I(f))^2\,dt\right)^{1/2}. \tag{2.9}$$

**Remark 2.1**

The expected error of the Monte Carlo algorithm for the function $f$ depends on its variance $\int_T (f(t)-I(f))^2\,dt$. There are many techniques on how to decrease the variance by performing certain transformations on the function $f$, see e.g., Davis and Rabinowitz [84] for a survey. These techniques require some additional information about the function $f$ such as extra smoothness, a particular form, or location of singularities. In our case, we assume that the only information about $f$ is given by its function values and by the fact that $f$ belongs to $F$. Therefore, in general, techniques for decreasing the variance are not applicable under our assumptions. $\Box$

Take now the supremum in (2.9) with respect to functions $f$ from $F = BL_2(T)$ or $F = C_d$. Then

$$e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F, \lambda) = \Theta(n^{-1/2}), \tag{2.10}$$

where the constants in the $\Theta$ notation do not depend on $d$.

Thus the Monte Carlo algorithm converges even for non-smooth functions. Moreover, the rate of convergence does not depend on the dimension $d$.

It is natural to ask if the error of the Monte Carlo algorithm can be improved by the use of different distributions of sample points or by differently combining the computed function values. That is, let $\Phi_n$ denote the class of all (possibly randomized) algorithms that use function values at randomized points $f(t_1), f(t_2), \dots, f(t_{n(f)})$ with respect to some distribution $\rho$. Here, $n(f)$ is also a random variable with expected value at most $n$. Let

$$e_n^{\mathrm{ran}}(F) = \inf_{\phi\in\Phi_n} e^{\mathrm{ran}}(\phi, F) \tag{2.11}$$

denote the $n$th minimal error of such algorithms.

Bakhvalov [59] for $n(f) \equiv n$, and Novak [88] for randomized $n(f)$ proved that

$$e_n^{\mathrm{ran}}(C_d) = \Theta(n^{-1/2}), \quad e_n^{\mathrm{ran}}(C_d^{r,\alpha}) = \Theta(n^{-(\frac{r+\alpha}{d}+\frac{1}{2})}). \tag{2.12}$$

This means that the Monte Carlo algorithm is optimal (to within a multiplicative constant) for the class $C_d$ and $BL_2(T)$, and is close to optimal for the class $C_d^{r,\alpha}$ whenever $d$ is large relative to $r+\alpha$.

The exponents in the $n$th minimal errors for the deterministic and randomized cases differ by 1/2. This is essential. To see this, compare the smallest numbers of function evaluations necessary to guarantee that the $n$th minimal errors do not exceed $\varepsilon$. For the randomized case and the class $C_d^{r,\alpha}$ we have

$$n^{\mathrm{ran}}(\varepsilon) = \Theta(\varepsilon^{-2/(1+2(r+\alpha)/d)}) = O(\varepsilon^{-2}). \tag{2.13}$$

Thus, $n^{\mathrm{ran}}(\cdot)$ is at most a quadratic function in $\varepsilon^{-1}$ for all $d$, as opposed to an exponential function $n^{\mathrm{det}}(\cdot)$ of $d$ in the deterministic case, see (2.6).

## 3. Lower Bounds on the Error of the Monte Carlo Algorithm with Pseudo-random Number Generators

The good properties of the Monte Carlo algorithm are based on two assumptions:

(1) the points $t_1, t_2, \ldots, t_n$ are independent and uniformly distributed random variables over $T = [0,1]^d$,

(2) the effect of randomness is measured by the expected distance between $I(f)$ and $\frac{1}{n}\sum_{i=1}^n f(t_i)$ with respect to the points $t_i$.

In computational practice, the points $t_i$ are selected by using a pseudo-random generator. This generator produces a sequence of numbers from which the points $t_i$ are calculated. Clearly, any pseudo-random generator can at best approximate independent and uniformly distributed random variables. It is natural to ask whether the good error properties of the Monte Carlo algorithm can be preserved when a pseudo-random generator is used.

Technically, this corresponds to the problem of analyzing the Monte Carlo algorithm if the assumption (i) is only approximately satisfied whereas the assumption (ii) is left unchanged. Assume thus that the points $\vec{t} = [t_1, t_2, \ldots, t_n] \in T^n$ are drawn with respect to some distribution $\rho$. The expected error of the Monte Carlo algorithm for a function $f$ is now defined as

$$E_n(f,\rho) = \left( \int_{T^n} \left( I(f) - \frac{1}{n}\sum_{i=1}^n f(t_i) \right)^2 \rho(d\vec{t}) \right)^{1/2}. \tag{3.1}$$

By the maximal expected error in the class $F$ we mean, compare with (2.8),

$$e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F, \rho) = \sup_{f \in F} E_n(f,\rho). \tag{3.2}$$

Assume that a pseudo-random generator can produce only finitely many different numbers from which the points $t_i$ are calculated. Then the distribution $\rho$ is an atomic measure supported on the points, say, $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_k$ for some $k$, usually very large, where $\vec{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$ with $x_{i,j} \in T$. The number $k$ may or may not depend on the dimension $d$. This depends on how pseudo-random numbers are used to compute the points $\vec{x}_i$. We then have for any function $H : T^n \to \mathbb{R}$,

$$\int_{T^n} H(\vec{x})\, \rho(d\vec{x}) = \sum_{i=1}^k p_i H(\vec{x}_i) \tag{3.3}$$

for some nonnegative $p_i$ and $\sum_{i=1}^k p_i = 1$.

Let $k^*$ denote the total number of different components of the points $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_k$. That is, $k^*$ is the cardinality of the set

$$\{x_{i,j} \in T : i = 1, 2, \ldots, k, \ j = 1, 2, \ldots, n\} = \{t_1^*, t_2^*, \ldots, t_{k^*}^*\}.$$

Clearly, $k^* \leq kn$. As we shall see later, depending on the way how the points $t_i^*$ are computed from pseudo-random numbers, $k^*$ may depend on $d$ and $n$, or $k^*$ may be independent of $d$ and $n$.

The form (3.3) of $\rho$ enables us to find a lower bound on the expected error $E_n(f, \rho)$ for some functions $f$ from the class $F$. To accomplish this, take a function $f$ from $F$ which vanishes at all the points $t_1^*, t_2^*, \ldots, t_{k^*}^*$. Then $E_n(f, \rho) = |I(f)|$. If $F$ is convex and balanced, i.e., $f \in F$ implies that $-f \in F$, then it is known that

$$\sup\{|I(f)| \ : \ f \in F, \ f(t_i^*) = 0, \ 1 = 1, 2, \ldots, k^*\} \ = \ e_{k^*}^{\mathrm{det}}(F),$$

where, as in (2.2), $e_{k^*}^{\mathrm{det}}(F)$ denotes the $k^*$th deterministic minimal error which can be achieved by using $k^*$ function values. Thus,

$$e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F, \rho) \ \geq \ e_{k^*}^{\mathrm{det}}(F). \tag{3.4}$$

For some convex and balanced classes $F$, the $k^*$ deterministic minimal error is constant. For example, if $F = BL_2(T)$ or $F = C_d$ then $e_{k^*}^{\mathrm{det}}(F) = 1$, $\forall k^*$, see (2.3). Therefore we have

**Theorem 3.1.** *The Monte Carlo algorithm with an arbitrary pseudo-random generator which is capable of producing only finitely many points does not converge for the class* $F = BL_2(T)$ *or* $F = C_d$*,*

$$e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F, \rho) \ \geq \ 1 \quad \text{for } F = BL_2(T) \text{ or } F = C_d. \tag{3.5}$$

Theorem 3.1 should be contrasted with (2.10) which states that for $\rho = \lambda$ (the Lebesgue measure) we obtain the Monte Carlo algorithm whose convergence is $e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F, \lambda) = \Theta(n^{-1/2})$.

**Remark 3.1**

The estimate (3.5) remains true for an arbitrary algorithm $\phi(f(t_1), f(t_2), \ldots, f(t_n))$. That is, for $F = BL_2(T)$ or $F = C_d$ we still have

$$\inf_{\phi} \sup_{f \in F} \left( \int_{T^n} \left(I(f) - \phi(f(t_1), f(t_2), \ldots, f(t_n))\right)^2 \rho(\vec{t}) \right)^{1/2} \geq 1. \quad \square$$

This negative result can be interpreted as stating that the use of a pseudo-random generator resulting in finitely many points $t_i^*$ requires more smoothness of $f$ than the Monte Carlo algorithm.

On the other hand, if we assume sufficient smoothness of functions then even the deterministic estimates become quite acceptable. For example, consider the class $F = C_d^{r,\alpha}$ with $r+\alpha$ comparable to $d$. Then the estimate (2.5) indicates that multivariate integration is a tractable problem.

Recall that the Monte Carlo algorithm works well even for non-smooth functions. That's why, using a pseudo-random generator, we would like to preserve this good property with as little extra smoothness of the functions $f$ as possible.

Clearly, continuity of $f$ is not enough. We compromise by assuming that $f$ satisfies a Lipschitz condition with constant, say, 1. Without loss of generality, we also assume that $f(0) = 0$. That is, let $f$ belongs to $F_d$, where

$$F_d = \{ f : T \to \mathbb{R} \, : \, f(0) = 0 \text{ and } |f(x) - f(y)| \le ||x - y||_\infty = \max_{1 \le i \le d} |x_i - y_i| \, \}. \tag{3.6}$$

This class has been analyzed in the deterministic worst case setting by Sukharev [79] who proved that

$$e_{k^*}^{\det}(F_d) \ge \frac{d}{2(d+1)} \lfloor (k^*)^{1/d} \rfloor^{-1} \tag{3.7}$$

and we have equality in (3.7) if $(k^*)^{1/d}$ is an integer. Using (3.7) and (3.4) we conclude that

$$e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F_d, \rho) \ge \frac{d}{2(d+1)} \lfloor (k^*)^{1/d} \rfloor^{-1}. \tag{3.8}$$

As mentioned before, $k^*$ may or may not depend on $d$ and $n$. If $k^*$ does *not* depend on $d$ and $n$ then for large $d$, the right-hand side of (3.8) is close to $1/2$. Furthermore, since $\lfloor (k^*)^{1/d} \rfloor^{-1}$ goes to 1 very quickly as $d$ increases, even for very large $k^*$ and modest $d$, the right-hand side of (3.8) is not small. For example, take $k^* = 2^{30}$ and $d = 10$. Then $\frac{d}{2(d+1)} \lfloor (k^*)^{1/d} \rfloor^{-1} = \frac{5}{88}$. In this case, we again lose the convergence rate enjoyed by the Monte Carlo algorithm. We now present an example where this happens.

**Example 3.1**

One of the most widely used pseudo-random generators is a linear congruential generator proposed by Lehmer [49] and extensively studied in the literature, see e.g., Niederreiter [78] and Knuth [81]. The linear congruential generator produces a sequence of integers

$$\begin{aligned} a_0 &= \text{an initial seed which is an integer from } [0,\text{m}) \\ a_{i+1} &= (\lambda\, a_i + r) \bmod m\,, \quad i = 0, 1, \ldots . \end{aligned}$$

Here, the modulus $m$ is taken as a large prime or a large power of 2, the multiplier $\lambda$ is a positive integer relatively prime to $m$, and $r$ is called the increment. The parameters $m, \lambda$ and $r$ have to be chosen such that the sequence $\{a_i\}$ passes appropriate statistical tests. In particular, the period of this sequence should be about $m$.

From the integers $a_i$ one can compute the points $x_i$ as folows. Let $z_i = a_i/m$ and

$$x_i = [z_i, z_{i+1}, \dots, z_{i+d-1}].$$

The properties and shortcomings of the points $x_i$ have been extensively studied and surveys may be found in the works mentioned above. Note that all the points $x_i$ are fully determined by the initial seed $a_0$. To stress this dependence we write $x_i = x_i(a_0)$. The points $\vec{x} = \vec{x}(a_0)$ are then given by

$$\vec{x}(a_0) = [x_1(a_0), x_2(a_0), \dots, x_n(a_0)] \in T^{nd}.$$

By changing the initial seed $a_0$ we obtain different points $\vec{x}(a_0)$. Since $a_0$ can take at most $m$ different values, we have $k \le m$ different points $\vec{x}(a_0)$. The distribution $\rho$ takes now the form, see (3.3),

$$\int_{T^n} H(\vec{x}\,)\,\rho(d\vec{x}\,) = \sum_{0 \le a_0 \le m-1} p_{a_0} H(\vec{x}(a_0))$$

for some weights $p_{a_0}$. (If $a_0$ takes $m$ different values with equal probability then $p_{a_0} = 1/m$.)

Now, the number of different components of $\vec{x}(a_0)$'s is equal to the number of different points $x_1(a_0), x_2(a_0), \dots, x_n(a_0)$. Clearly, there are at most as many such points as there are different values of $a_0$. Thus, there are at most $m$ of them. Hence, $k^* \le m$. In this case, $k^*$ does not depend on $d$ and $n$, and we may apply the estimate (3.8). This proves that this use of the linear congruential generator leads to poor results for the class $F_d$ with large $d$. $\square$

**Remark 3.2**

It should be noted that the negative result of Example 3.1 does not hold if one assumes extra smoothness of functions $f$. More precisely, assuming that $f$ has a finite variation $V(f)$ in the sense of Hardy and Krause then the Koksma-Hlavka inequality states that

$$\Big|I(f) - \frac{1}{n}\sum_{i=1}^{n} f(x_i)\Big| \le V(f)\,D_n, \tag{3.9}$$

where $D_n$ is the discrepancy of the points $x_1, x_2, \dots, x_n$; see Niederreiter [78, 88, 89] for excellent surveys on this subject.

The total variation $V(f)$ is finite if $f$ is once differentiable in each direction $t_i$, and $\partial^j f/\partial t_{i_1} \dots \partial t_{i_j}$ for $j \le d$ belongs to $L_1(T)$. (Therefore not all functions in $F_d$ have finite variation !)

For the points $x_i$ given by a linear congruential generator, the discrepancy $D_n$ was studied by Niederreiter who proved, in particular, that for a good choice of the parameters we have

$$D_n = O\left(\frac{\sqrt{m}(\log m)^{d+1}}{n}\right), \quad \text{for } n \le m,$$

see Niederreiter [78, p.1018].

The estimate (3.9) suggests that as long as we consider functions of finite variation, we should choose points which minimize the discrepancy $D_n$. This problem has been also extensively studied and Halton [60] proved that there exist points for which the discrepancy $D_n$ is of order $n^{-1}(\log n)^{d-1}$; see Niederreiter [78, 88, 89] for a survey what is known about the discrepancy.

In summary, the extra smoothness introduced by the existence of finite variation makes multivariate integration a tractable problem since with a good choice of $n$ deterministic points the error goes to zero as $n^{-1}(\log n)^{d-1}$. Hence, the rate of convergence depends very mildly on the dimension $d$. For such a class of functions, we obviously shouldn't use randomization. □

The discussion above indicates that for the class $F_d$ we should use a pseudo-random generator and calculate the points $t_i^*$ in such a way that $k^*$ depends on $d$ and/or $n$ and is as large as possible. In view of the right-hand side of (3.8), we would ideally like to have $\lfloor (k^*)^{1/d} \rfloor^{-1}$ be independent of $d$ and hopefully small. We now indicate how this can be achieved for two examples still using a linear congruential generator.

**Example 3.2**

As in Example 3.1, consider a linear congruential generator producing the integer sequence $\{a_i\}$ of period $m$. Suppose we use this generator $d$ times for each component of $x = [x_1(a_0^1), x_2(a_0^2), \dots, x_d(a_0^d)]$ with *independent* initial seeds $a_0^1, a_0^2, \dots, a_0^d$. Since each $x_i(a_0^i)$ can take $m$ different values and because they are independent, we could generate $m^d$ different points of $T$. Thus, $k^* = m^d$ and (3.8) becomes now

$$e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F_d, \rho) \ge \frac{d}{2(d+1)} \frac{1}{m}. \tag{3.10}$$

Since $m$ is usually huge, the right-hand side of (3.10) is quite satisfactory. □

**Example 3.3**

For simplicity assume that $n = p^d$ and let $h = 1/p$. Partition the cube $T$ into $n$ subcubes

$$T_{i_1,i_2,\ldots,i_d} = [i_1 h, (i_1+1)h) \times [i_2 h, (i_2+1)h) \times \cdots \times [i_d h, (i_d+1)h)$$

for $i_1, i_2, \ldots, i_d = 0, 1, \ldots, p-1$.

Using a linear congruential generator of period $m$ select *only* one vector

$$x = [x_1(a_0), x_2(a_0), \ldots, x_n(a_0)].$$

Then define the points $\vec{x}_{i_1,\ldots,i_d}$ by shifting $h\,x$ to each subcube $T_{i_1,\ldots,i_d}$, i.e.,

$$\vec{x}_{i_1,\ldots,i_d} = [i_1 h, \ldots, i_d h] \,+\, h\,x.$$

Then we will have $k^* = m\,n$ different points $\vec{x}_{i_1,\ldots,i_d}$, and (3.8) becomes

$$e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F_d, \rho) \geq \frac{d}{2(d+1)}\, \lfloor (m\,n)^{1/d} \rfloor^{-1}.$$

Although, the lower bound goes now to zero as $n$ goes to $\infty$, we still have a bad dependence on $d$. This can be improved by running the linear congruential generator $d$ times for each component of $x$ independently, as in Example 3.2. That is, we now have

$$\begin{aligned} x &= [x_1(a_0^1), x_2(a_0^2), \ldots, x_d(a_0^d)], \\ \vec{x}_{i_1,\ldots,i_d} &= [i_1 h, \ldots, i_d h] \,+\, h\,x. \end{aligned}$$

In this case, we have $k^* = m^d\,n$ different points $\vec{x}_{i_1,\ldots,i_d}$ and (3.8) becomes

$$e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F_d, \rho) \geq \frac{d}{2(d+1)}\, \frac{1}{m\,n^{1/d}}.$$

The lower bound is now always small and goes to zero, although very slowly, as $n$ tends to infinity. $\square$

The essence of Examples 3.2 and 3.3 is that the lower bound on the error is small if a linear congruential generator is used for each component independently. In the next section we derive general upper bounds on the expected error of the Monte Carlo algorithm with a distribution $\rho$. We then apply these bounds to linear congruential generators as in Examples 3.2 and 3.3.

## 4. Upper Bounds on the Error of the Monte Carlo Algorithm with Pseudo-Random Generators

In this section we analyze the Monte Carlo algorithm with points $t_i$ which are produced by pseudo-random generators such that the $t_i$ are independent random variables and each $t_i$ is chosen with respect to some distribution $\rho_i$ which is not necessarily a Lebesgue measure.

Let $\rho = [\rho_1, \rho_2, \dots, \rho_n]$. Then (3.1) takes the form

$$E_n^2(f,\rho) = \int_{T^n} \left( I(f) - \frac{1}{n} \sum_{i=1}^n f(t_i) \right)^2 \rho_1(dt_1) \cdots \rho_n(dt_n). \tag{4.1}$$

In order to analyze $E_n(f,\rho)$, define the error function

$$e(f,\rho_i) = \int_T f(t)\,dt \;-\; \int_T f(t)\,\rho_i(dt) \tag{4.2}$$

as the difference between the integrals of $f$ with respect to the Lebesgue and $\rho_i$ measures. Let

$$\bar{\rho_n} = \frac{1}{n} \sum_{i=1}^n \rho_i \tag{4.3}$$

denote the arithmetic mean of the distributions $\rho_i$. Note that for $\rho_i = \lambda$, we have $\bar{\rho_n} = \lambda$ and $e(f,\rho_i) = e(f,\bar{\rho_n}) = 0$. We may expect that if $\rho_i$ is "close" to $\lambda$ then $e(f,\rho_i)$ and $e(f,\bar{\rho_n})$ should be "close" to zero.

**Lemma 4.1.** *If $f^2$ is integrable with respect to the measures $\lambda$ and $\rho_i$ then*

$$\begin{aligned} E_n^2(f,\rho) = {} & \frac{1}{n}\left(I(f^2) - I^2(f)\right) \;+\; \frac{1}{n}\left(2\,I(f)\,e(f,\bar{\rho_n}) \;-\; e(f^2,\bar{\rho_n})\right) \\ & + \; e^2(f,\bar{\rho_n}) \;-\; \frac{1}{n^2}\sum_{i=1}^n e^2(f,\rho_i). \quad \Box \end{aligned} \tag{4.4}$$

*Proof.* To show (4.4) observe that (4.1) yields

$$\begin{aligned} E_n^2(f,\rho) &= I^2(f) - \frac{2}{n} I(f) \sum_{i=1}^n \int_T f(t)\,\rho_i(dt) \\ &\quad + \frac{1}{n^2} \sum_{i,j=1}^n \int_{T^n} f(t_i) f(t_j)\,\rho_1(dt_1)\cdots\rho_n(dt_n) \\ &= I^2(f) - 2I(f) \int_T f(t)\,\bar{\rho_n}(dt) \\ &\quad + \frac{1}{n^2}\left( \sum_{i=1}^n \int_T f^2(t)\,\rho_i(dt) \;+\; \sum_{i \neq j} \int_T f(t)\,\rho_i(dt) \; \int_T f(t)\,\rho_j(dt) \right). \end{aligned}$$

Using the definition of $e(f,\cdot)$ and the indentity $\sum_{i\neq j} a_i a_j = (\sum_i a_i)^2 - \sum_i a_i^2$, we have

$$\begin{aligned}
E_n^2(f,\rho) &= I^2(f) - 2I(f)\,(I(f) - e(f,\bar{\rho_n})) + \frac{1}{n}\int_T f^2(t)\,\bar{\rho_n}(dt) \\
&+ \left(\int_T f(t)\bar{\rho_n}(dt)\right)^2 - \frac{1}{n^2}\sum_{i=1}^n \left(\int_T f(t)\,\rho_i(dt)\right)^2 \\
&= -I^2(f) + 2I(f)e(f,\bar{\rho_n}) + \frac{1}{n}I(f^2) - \frac{1}{n}e(f^2,\bar{\rho_n}) \\
&+ I^2(f) - 2I(f)e(f,\bar{\rho_n}) + e^2(f,\bar{\rho_n}) - \frac{1}{n^2}\sum_{i=1}^n (I(f) - e(f,\rho_i))^2 \\
&= \frac{1}{n}\left(I(f^2) - I^2(f)\right) + \frac{1}{n}\left(2I(f)e(f,\bar{\rho_n}) - e(f^2,\bar{\rho_n})\right) \\
&+ e^2(f,\bar{\rho_n}) - \frac{1}{n^2}\sum_{i=1}^n e^2(f,\rho_i),
\end{aligned}$$

as claimed in (4.4). □

We illustrate Lemma 4.1 by two examples.

**Example 4.1**

We check (4.4) for the Monte Carlo algorithm. As mentioned earlier, for $\rho_i = \lambda$ we have $e(\cdot,\rho_i) = e(\cdot,\bar{\rho_n}) = 0$, and (4.4) becomes

$$E_n(f,\rho) = \frac{1}{\sqrt{n}}\sqrt{I(f^2) - I^2(f)} = \frac{1}{\sqrt{n}}\sqrt{I\left((f - I(f))^2\right)}$$

which is the well known formula for the expected error of the Monte Carlo algorithm. □

**Example 4.2**

Assume that the arithmetic mean $\bar{\rho_n}$ is a Lebesgue measure $\lambda$ but $\rho_1,\ldots,\rho_n$ are not necessarily equal to $\lambda$. Then $e(\cdot,\bar{\rho_n}) = 0$, and

$$E_n(f,\rho) = \left(\frac{1}{n}\left(I(f^2) - I^2(f)\right) - \frac{1}{n^2}\sum_{i=1}^n e^2(f,\rho_i)\right)^{1/2}.$$

Observe that now the expected error $E_n(f,\rho)$ is no larger than the expected error for the uniform distribution.

The case $\bar{\rho_n} = \lambda$ can be achieved if, for example, the cube $T$ is partitioned into $n$ disjoint subcubes $T_i$, each of them of the same Lebesgue measure, and $\rho_i$ is defined as a uniform distribution over $T_i$, as was considered by Haber [66]. □

We now discuss (4.4). Since the last term in (4.4) is nonpositive we have

$$E_n^2(f,\rho) \le \frac{1}{n}\left(I(f^2) - I^2(f)\right) + \frac{1}{n}\left(2I(f)e(f,\bar{\rho_n}) - e(f^2,\bar{\rho_n})\right) + e^2(f,\bar{\rho_n}). \tag{4.5}$$

Note that the first term of (4.5) is the square of the expected error of the Monte Carlo algorithm, whereas the rest of the terms depend on the distribution $\bar{\rho_n}$ which is the arithmetic mean of the distributions $\rho_i$. If $e(f^2,\bar{\rho_n}) = O(1)$, then the second term is proportional to $n$, and we may rewrite (4.5) as

$$E_n(f,\rho) = O\left(\frac{1}{\sqrt{n}} + |e(f,\bar{\rho_n})|\right).$$

Thus, we can preserve the rate of convergence $n^{-1/2}$, if $e(f,\bar{\rho_n}) = O(n^{-1/2})$. We now show that this is the case for the class $F_d$, see (3.6), when we use a linear congruential generator $d$ times as described in Example 3.3.

**Example 4.3**

As in Example 3.3, let $n = p^d$ and $h = 1/p$. We partition the cube $T$ into $n$ subcubes $T_{i_1,\dots,i_d}$. A linear congruential generator of period $m$ is used $d$ times independently for each component. Assume that initial seeds are equiprobable. Then the distribution $\rho_{i_1,\dots,i_d}$ is a uniform distribution over the points $[i_1 h,\dots,i_d h] + m^{-1}h\,[j_1,\dots j_d]$ from $T_{i_1,\dots,i_d}$, $j_1,\dots,j_d = 0,1,\dots m-1$.

Then for the arithmetic mean $\bar{\rho_n}$ and for any integrable function $H$ we have

$$\begin{aligned}
\int_T H(t)\,\bar{\rho_n}(dt) &= \frac{1}{n}\sum_{0\le i_1,\dots,i_d\le p-1}\int_{T_{i_1,\dots,i_d}} H(t)\,\rho_{i_1,\dots,i_d}(dt)\\
&= \frac{1}{n\,m^d}\sum_{0\le i_1,\dots,i_d\le p-1}\ \sum_{0\le j_1,\dots,j_d\le m-1} H([i_1 h,\dots,i_d h] + m^{-1}h[j_1,\dots,j_d])\\
&= \tau^d \sum_{0\le i_1,\dots,i_d\le p\,m-1} H([i_1\tau,\dots,i_d\tau]),
\end{aligned}$$

where $\tau = 1/(p\,m)$.

We estimate $e(f,\bar{\rho_n})$ and $e(f^2,\bar{\rho_n})$ for functions $f$ from $F_d$. Let

$$U_{i_1,\dots,i_d} = [i_1\tau,(i_1+1)\tau)\times\cdots\times[i_d\tau,(i_d+1)\tau),\quad i_1,\dots,i_d = 0,1,\dots,p\,m-1.$$

We have

$$e(f,\bar{\rho_n}) = \sum_{0\le i_1,\dots,i_d\le p\,m-1}\int_{U_{i_1,\dots,i_d}} \left(f(t) - f([i_1\tau,\dots,i_d\tau])\right)\,dt.$$

Since $f$ satisfies a Lipschitz condition,

$$|f(t) - f([i_1\tau, \dots, i_d\tau])| \le ||t - [i_1\tau, \dots, i_d\tau]||_\infty \le \tau, \quad \forall t \in U_{i_1,\dots,i_d}.$$

Thus,

$$|e(f, \bar{\rho_n})| \le \sum_{0 \le i_1,\dots,i_d \le p\, m-1} \tau\, \lambda(U_{i_1,\dots,i_d}) \;=\; \tau \;=\; \frac{1}{m\, n^{1/d}}.$$

To estimate $e(f^2, \bar{\rho_n})$, note that $f \in F_d$ implies that $|f(t)| \le 1, \forall t \in T$. For $x = [i_1\tau, \dots, i_d\tau]$ we thus have

$$|f^2(t) - f^2(x)| \le |f(t) + f(x)|\, |f(t) - f(x)| \le 2\tau, \quad \forall t \in U_{i_1,\dots,i_d}.$$

This yields

$$e(f^2, \bar{\rho_n}) \le 2\tau \;=\; \frac{2}{m\, n^{1/d}}.$$

Since $|I(f)| \le 1$ for $f \in F_d$, we get from (4.5)

$$\left|E_n^2(f,\rho) - \frac{1}{n}\left(I(f^2) - I^2(f)\right)\right| \le \frac{1}{n}\,\frac{4}{m\, n^{1/d}} \;+\; \frac{1}{m^2\, n^{2/d}},$$

$$\left|E_n(f,\rho)\right| \le \frac{1}{\sqrt{n}}\,\sqrt{1 + \frac{4}{m\, n^{1/d}} + \frac{n^{1-2/d}}{m^2}}.$$

Hence, as long as $n^{1-2/d}$ is at most of order $m^2$, the error of the Monte Carlo algorithm with distribution $\rho$ is of order $n^{-1/2}$ as is the error of the Monte Carlo algorithm.

We stress that the inequality $n^{1-2/d} \le m^2$ is not restrictive in practice. Indeed, a typical value of $m$ is $2^{30}$ and for large $d$, even $n \le m^2$ means $n \le 2^{60}$.

We summarize upper and lower bounds on $e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F_d, \rho) = \sup_{f \in F_d} E_n(f,\rho)$, from this Example and Example 3.3 in

**Theorem 4.1.** *For the class $F_d$ and $n \le m^2$, the Monte Carlo algorithm with independent points produced by a linear congruential generator of period $m$ with uniformly distributed initial seeds applied independently to each component with the shift to the subcubes $T_{i_1,\dots,i_d}$ behaves essentially as the Monte Carlo algorithm,*

$$\frac{d}{2(d+1)}\,\frac{1}{m\, n^{1/d}} \le e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F_d, \rho) \le \frac{1}{\sqrt{n}}\,\sqrt{1 + \frac{4}{m\, n^{1/d}} + \frac{n^{1-2/d}}{m^2}}. \quad \square$$

Finally, we discuss the case for which all the points $t_i$ are indentically distributed, $\rho_i = \rho$. Then $\bar{\rho_n} = \rho$ and (4.4) takes the form

$$E_n^2(f,\rho) = \frac{1}{n}\left(I(f^2) - I^2(f)\right) + \frac{1}{n}\left(2I(f)e(f,\rho) - e(f^2,\rho)\right) + \left(1 - \frac{1}{n}\right) e^2(f,\rho). \qquad (4.6)$$

The last equality can be rewritten as

$$E_n(f,\rho) = O\left(\frac{1}{\sqrt{n}} + |e(f,\rho)|\right).$$

Thus, as long as $e(f,\rho)$ is at most of order $n^{-1/2}$, the expected error behaves as for the uniform distribution. On the other hand, if $n$ goes to $\infty$, then

$$\lim_n E_n(f,\rho) = |e(f,\rho)|.$$

This shows that if $\rho$ is not the uniform distribution then the expected error does not, in general, tend to zero. If, however, $\rho$ is "close" to the uniform distribution then the limit of $E_n(f,\rho)$ is small. We illustrate this point by two examples.

**Example 4.4**

Assume that $\rho$ is absolutely continuous to $\lambda$,

$$\rho(A) = \int_A h(t)\,dt, \quad \text{for any Borel set } A,$$

with a weight function $h : T \to \mathbb{R}_+$. Then

$$e^2(f,\rho) = \left(\int_T f(t)\,(1-h(t))\;dt\right)^2 \le I^2(f)\int_T (1-h(t))^2\;dt.$$

Thus, if $h$ is a good approximation of the function which is indentically equal to 1, then the limit of $E_n(f,\rho)$ is small. $\square$

**Example 4.5**

Assume, as in Example 3.2, that a linear congruential generator of period $m$ is used $d$ times for each component independently. Assume that initial seeds are equiprobable. Then the distribution $\rho$ is a uniform distribution over the points $\{i_1h, i_2h, \dots, i_dh\}$ for $i_1,\dots,i_d = 0,\dots,m$ with $h = 1/m$. (Note that $\rho$ is *not* absolutely continuous with respect to $\lambda$.)

We estimate $e(f,\rho)$ and $e(f^2,\rho)$ as in Example 4.3 and obtain

$$|e(f,\rho)| \le \frac{1}{m}, \quad |e(f^2,\rho)| \le \frac{2}{m}.$$

From these estimates we conclude that

$$|E_n^2(f,\rho) - \frac{1}{n}\left(I(f^2) - I^2(f)\right)| \le \frac{4}{m\,n} + \frac{1}{m^2},$$

$$|E_n(f,\rho)| \le \frac{1}{\sqrt{n}}\sqrt{1 + \frac{4}{m} + \frac{n}{m^2}}.$$

Observe that the bounds above differ from the bounds of Example 4.3 by the factor $n^{1/d}$. For large $d$ it is not essential.

Hence, as long as $n \le m^2$, the error of the Monte Carlo algorithm with the distribution $\rho$ is of order $n^{-1/2}$ as is the error of the Monte Carlo algorithm. We summarize upper and lower bounds of this Example and Example 3.2 in

**Theorem 4.2.** *For the class $F_d$ and $n \le m^2$, the Monte Carlo algorithm with independent points produced by a linear congruential generator of period $m$ with uniformly distributed initial seeds applied independently to each component behaves essentially as the Monte Carlo algorithm,*

$$\frac{d}{2(d+1)}\,\frac{1}{m} \le e^{\mathrm{ran}}(\phi_n^{\mathrm{MC}}, F_d, \rho) \le \frac{1}{\sqrt{n}}\sqrt{1+\frac{4}{m}+\frac{n}{m^2}}. \quad \square$$

## 5. Open Problems

**1.** The emphasis in this paper is on pseudo-random generators which are capable of producing only finitely many points. In particular, a linear congruential generators falls in this category. It would be interesting to extend the analysis of this paper to other generators which, at least theoretically, produce infinitely many points. An example of such generator is provided by $z_n = \theta^n \bmod 1$, where $\theta$ is a transcendental number greater than 1, see Franklin's theorem in Knuth [81, p. 152]. Observe that for such generators the lower bound (3.8) is trivially zero. An open problem is which of these generators preserve the good properties of the Monte Carlo algorithm for the class of, say, Lipschitz functions.

**2.** We have analyzed the class of Lipschitz functions with constant 1. As mentioned in Section 2, there exists a randomized algorithm $\phi$ which, for this class, converges at rate $n^{-1/2-1/d}$. The Monte Carlo algorithm does not have an optimal rate of convergence since its error is proportional to $n^{-1/2}$. Of course, for large $d$ the difference in the exponent is not significant. Nevertheless, it would be interesting to analyze the algorithm $\phi$ with a pseudo-random generator and to check under what conditions we can expect an error of order $n^{-1/2-1/d}$.

**3.** In this paper we studied integration for the class of Lipschitz functions. Different classes with more smoothness should also be analyzed. For example, classes $C_d^{r,\alpha}$ look like natural candidates. What kind of pseudo-random generators should be used to preserve the error estimates (2.12)?

**4.** We restrict ourselves in this paper to multivariate integration. It is known that for some other problems randomization also helps significantly. For example, we mention two such problems. The first one is weighted multivariate integration for which non-uniform distributions should be used. For which pseudo-random generators is it possible to preserve optimal rate of convergence for weighted multivariate integration?

The second problem is an algebraic eigenvalue problem for which we wish to approximate the largest eigenvalue of a large $n \times n$ symmetric positive definite matrix $A$ using matrix-vector multiplications. Using a random vector $b$ with uniform distribution over the unit sphere, one can compute Krylov information $Ab, A^2, \dots, A^k b$ and apply the Lanczos algorithm to get an approximation to the largest eigenvalue with relative error $O((\ln(n)/k)^2)$, see Kuczyński and Woźniakowski [89]. For which pseudo-random generators is it possible to preserve this error bound?

## References


Bach, E. *Realistic Analysis of Some Randomized Algorithms*, Proceedings of the 19th ACM Symposium on Theory of Computing (1987), 453–461.

Bakhvalov, N. S. *On approximate computation of integrals (in Russian)*, Vestnik MGV, Ser. Math. Mech. Astron. Phys. Chem. **4** (1959), 3–18.

Blum, M. and Micali, S. *How to generate cryptographically strong sequences of pseudo-random bits*, SIAM Journal on Computing **13** (1984), 850–863.

Davis, P. J. and Rabinowitz, P. *Methods of numerical integration*, Academic Press, New York, 1984.

Haber, S. *A modified Monte Carlo quadrature*, Math. Comp. **20** (1966), 361–368.

Halton, J. H. *On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals*, Numer. Math. **2** (1960), 84–90.

Hammersley, J. M. and Handscomb, D. C. *Monte Carlo Methods*, Methuen, London,, 1964..

Granovskii, B. L. and Ermakov, S. M. *The Monte Carlo method*, J. Soviet Math. **7** (1977), 161–192.

Kalos, P. A. and Whitlock, P. A. *Monte Carlo methods*, vol. 1, Wiley, New York, 1986.

Karloff, H. J. and Raghavan, P. *Randomized Algorithms and Pseudorandom Numbers*, Proceedings of the 20th ACM Symposium on Theory of Computing (1988), 310–321.

Knuth, D. E. *The Art of Computer Programming*, vol. 2 Second Edition, Addison Wesley, Reading, Mass., 1981.

Kuczyński, J. and Woźniakowski, H. *Estimating the Largest Eigenvalue by the Power and Lanczos Algorithms with a Random Start*, Report, Dept. of Computer Science, Columbia University, 1989..

Lehmer, D. H. *Mathematical models in large-scale computing units*, Proc. 2nd Sympos. on Large-Scale Digital Calculating Machinery (Cambridge, Mass., 1949) (1951), 141–146, Harvard Univ. Press, Cambridge, Mass..

Niederreiter, H. *Quasi-Monte Carlo methods and pseudo-random numbers*, Bull. Amer. Math. Soc. **84** (1978), 957–1041.

Niederreiter, H. *Quasi-Monte Carlo methods for multidimensional numerical integration*, in Numerical Integration III, International Series of Numerical Mathematics **85**, 157–171 (H. Braß and G. Hämmerlin eds.), Birhäuser Verlag, Basel, 1988.

Niederreiter, H. *Recent Trends in Random Number and Random Vector Generation*, Fifth Internat. Conf. on Stochastic Programming (1989), Ann Arbor, Mich.,.

Novak, E. *Deterministic and Stochastic Error Bounds in Numerical Analysis*, vol. 1349, Lectures Notes in Math. Springer Verlag, Berlin, 1988.

Sobol, I. M. *Die Monte Carlo-Methode*, Deutsch Verlag, Frankfurt, 1985.

Sukharev, A. G. *Optimal numerical integration formulas for some classes of functions of several variables*, Sov. Math. Dokl. **20** (1979), 472–475.

Traub, J. F., Wasilkowski, G. W. and Woźniakowski, H. *Information-Based Complexity*, Academic Press, New York, 1988.

Yao, A. *Theory and application of trapdoor functions*, Proceedings of the 23rd IEEE Symposium on Foundation of Computer Science (1982).