ELSEVIER

Mathematics and Computers in Simulation 46 (1998) 485–505

MATHEMATICS AND COMPUTERS IN SIMULATION

Good random number generators are (not so) easy to find

P. Hellekalek[*]

Dept. of Mathematics, Salzburg University, Hellbrunner Straße 34, A-5020 Salzburg, Austria

Abstract

Every random number generator has its advantages and deficiencies. There are no "safe" generators. The practitioner's problem is how to decide which random number generator will suit his needs best. In this paper, we will discuss criteria for good random number generators: theoretical support, empirical evidence and practical aspects. We will study several recent algorithms that perform better than most generators in actual use. We will compare the different methods and supply numerical results as well as selected pointers and links to important literature and other sources. Additional information on random number generation, including the code of most algorithms discussed in this paper is available from our web-server under the address **http://random.mat.sbg.ac.at/** © 1998 IMACS/Elsevier Science B.V.

1. Introduction

Random number generators ("RNGs") are the basic tools of stochastic modeling. As any other craftsman, the modeler has to know his tools. Bad random number generators may ruin a simulation. There are several pitfalls to be avoided.

For example, if we try to check the correlations between consecutive random numbers $x_0, x_1, \ldots$, then a (still!) widely used generator produces non-overlapping pairs (x_{2n}, x_{2n+1}), $n=0,1,\ldots$ above suspicion at the first look, see Fig. 1. In sharp contrast, the triples $(x_{3n}, x_{3n+1}, x_{3n+2})$ are extremely correlated and happen to lie on only fifteen planes, see Fig. 2. In both figures, 2^{15} points have been generated. For details on this phenomenon, we refer to Section 4 and Section 5.

In this paper, safety-measures against such unpleasant surprises will be given. We will discuss the current standards for good random number generators, the underlying mathematical and statistical concepts, and some new generators that meet these standards. Further, we will summarize the advantages and deficiencies of several algorithms to generate and test random numbers. Finally, we will present a "RNG Survival Kit" that contains the most important literature on this subject and links to web-sites that offer code and documents, and a "RNG Checklist" that allows the reader to assess his preferred generator on the basis of the concepts given in this concise survey.

[*] Corresponding author. E-mail: peter.hellekalek@sbg.ac.at

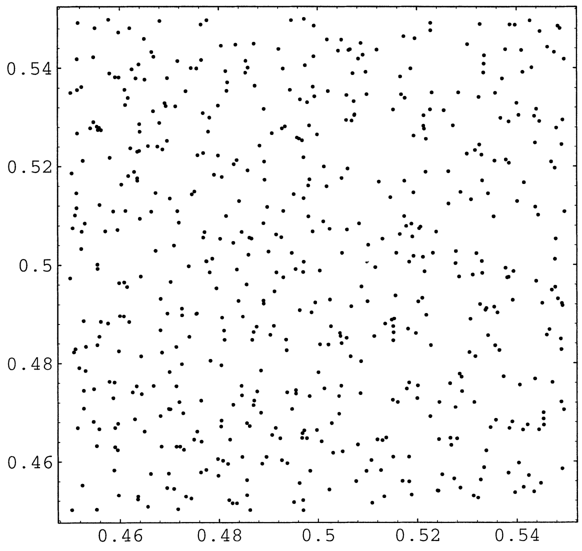

Fig. 1. LCG(2^{31}, 65539, 0, 1) Dimension 2: Zoom into the unit interval.

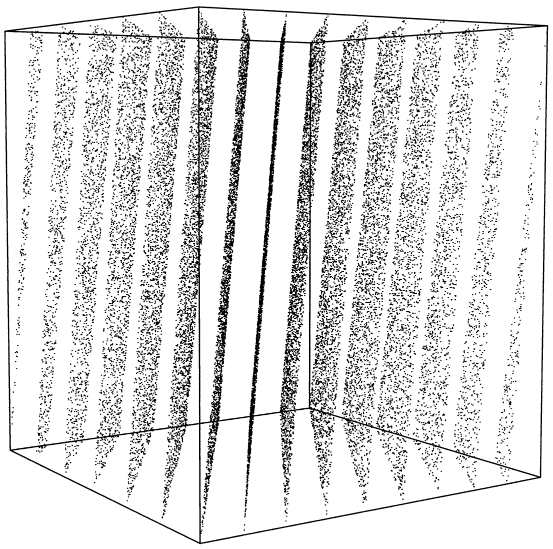

Fig. 2. LCG(2^{31}, 65539, 0, 1) Dimension 3: The 15 planes.

A good generator is not so easy to find if one sets out to design it by oneself, without the necessary mathematical background. On the other hand, with the references and links we supply, a good random number generator designed by experts is relatively easy to find.

The standard approach to generate nonuniform random numbers is to produce uniform random numbers first and then to transform them, see the monograph [7] and the software package CRAND [59,60]. In this paper, we will restrict our attention to uniform random number generators.

2. What is a good RNG?

The underlying problem is the following (see [2] for this quotation):

"Monte Carlo results are misleading when correlations hidden in the random numbers and in the simulated system interfere constructively."

The answer to the question above will depend on the target application. In the present paper, we will focus our interest on uniform random number generators appropriate for stochastic simulation. In cryptology, the requirements for generators are different, see [27].

From a practitioner's point of view, random number generators are good if they yield the correct results in as many applications as possible. This desire cannot be fulfilled completely. It is known that every generator has to fail in certain simulations, in models that interfere with the particular regularities of a given generator and exhibit the hidden correlations between the random numbers. This unpleasant situation is a fact of (scientific) life and cannot be circumvented, see [2,34], and [40]. Random number generators are nothing more than deterministic algorithms that produce numbers with certain distribution properties. These sequences of so-called "random" numbers are periodic. Their task is not to simulate "randomness", which is a notion that is difficult to define in terms of practical relevance, but to give the correct results in a simulation, and hence, to pass certain tests that the user considers relevant for his problem. The difficulty with randomness is partly due to the fact that random variables are often mixed up with numbers. For example, the notion of independence is only defined for random variables, not for numbers. We refer the reader to [63] for details and some enlightening comments on this subject.

How can we provide good random number generators if we don't know the target application in advance? The designer of a generator does not know the sampling procedure nor the sample sizes nor the dimensions the practitioner will choose in his simulation. There are two facts which every user of random number generators should know.

FACT ONE: With random number generators, there are *no guarantees, only predictions*. This is not because the word "randomness" is involved but because the finitely many random numbers we produce and their transformed variates cannot fit every imaginable distribution well enough. Every generator has its regularities which, ocassionally, may become deficiencies. Hence, in a given application, even reliable generators may fail.

FACT TWO: Although there are no guarantees, there are *mathematical safety-measures* against wrong simulation results caused by inappropriate random number generators.

The first precaution is *empirical testing* of random number generators. We may run *application-specific tests* with known theoretical results and compare them to our empirical results, see [62] for examples from physics. As an alternative, we may search in the existing literature for tests that resemble our simulation problem. If we are lucky, a particular test design and parameters like the sample size or the dimension will be similar to our setup. Frequently, we will be unlucky. Good starting points for our search are the surveys [53,55,31,34] and the "Links"-page at the Web-site **http://random.mat.sbg.ac.at/**

If we cannot find a test that resembles our simulation problem, then we may submit the random numbers we want to use to a battery of empirical tests. It should be noted that empirical tests cannot prove anything formally, like "randomness" for a random number generator. The only conclusion we can derive from the results of an empirical test is that the samples that have been used pass or fail this particular test, nothing more and nothing less. Nothing can be concluded for other samples, other dimensions, or other initializations of the generator under study. Of course, if our generator passes many empirical tests, this will improve our confidence as well as our chances to get the right simulation results with this generator. We refer to Section 5 for further information.

The second safety-measure is theoretical support for a random number generator. It means that we know about the period length of the generator, know some of its structural properties, and its correlation behavior, see Section 4 for details. The period length will limit the size of the samples we can use safely. The structural properties will help us to decide if there might be unwanted side-effects in the simulation. The correlation properties of a random number generator are of central importance for many stochastic simulations.

Practical aspects of random number generators concern the speed of the algorithm, the ease ofimplementation, the possibility to use parallelization techniques, and the availability of portable implementations. One generator of a given type will not be enough for numerical practice. We need tables of tested parameters to be able to implement several generators of the same kind. If we happen to work with high-performance computers, then we will also require that extremely large samples and, necessarily, large periods are available.

A good random number generator fulfills this catalog of safety standards. As a matter of fact, all available generators lack answers in certain sub-categories of this checklist. It is up to the practitioner to decide which aspects of a generator are important to him and to select an appropriate generator accordingly. We provide an "RNG Checklist" to assist with this choice in Section 9.

3. Examples of RNGs

In this section, we will present several recent advances in the construction of random number generators that merit a strong recommendation, in particular, to those practitioners that require large samples (i.e. long periods), speed, and/or little correlations. We will revisit these examples in Section 7, applying the concepts developed in Sections 4 and 5 and Section 6.

We refer the reader to the surveys [30,34] for a comprehensive discussion of linear algorithms. For a concise presentation of most of the available algorithms with an emphasis on the theoretical background and on nonlinear generators we recommend [55].

Linear methods are the best-known and most widely used algorithms to produce random numbers. Their practical advantages are speed, ease of implementation, and the availability of portable code, parameters and test results. An eye-catching phenomenon that occurs with linear types are lattice structures of the d-tuples constructed from consecutive random numbers. This fact is not at all a defect, but an intrinsic property of this family of generators. It allows to define and compute figures of merit like the *spectral test* or *Beyer quotients* (see Section 4).

The classical example of a random number generator is the linear congruential generator ("LCG"). It is defined by the linear congruence $y_{n+1}\equiv ay_n+b (\mathrm{mod}\ m)$, $n\geq 0$, where we have to choose the modulus m, a multiplier a, an additive term b, and an initial value y_0. This recursion of order one produces a sequence of numbers $(y_n)_{n\geq 0}$ in the set $\{0, 1, \ldots, m-1\}$. Random numbers x_n in [0,1] are obtained by the normalization $x_n:=y_n/m$. We will denote this generator by LCG(m, a, b, y_0). Examples of well-known LCGs are the Ansi-*C* system generator LCG(2^{31}, 1103515245, 12345, 12345), the "Minimal Standard" generator LCG($2^{31}-1$,16807, 0), infamous RANDU, which is LCG(2^{31}, 65539, 0), the SIMSCRIPT generator LCG($2^{31}-1$,630360016, 0), NAG's LCG(2^{59}, 13^{13}, 0, 123456789 ($2^{32}+1$)), and Maple's LCG($10^{12}-11$, 427419669081, 0, 1).

If we study the distribution of d-tuples $x_n=(x_n, x_{n+1},\ldots,x_{n+d-1})$ and if we generate all possible points in $[0,1[^d$, then we will observe lattice structures. This is a well-known phenomenon. In the examples below, all possible points in dimension $d=2$ have been produced.

Both LCGs have the period $2^{31}-2$, which is best possible in this case(see [[53], 7.3]). Figs. 3 and 4 shows that maximum period does not guarantee good lattice structure. It does not imply good correlation properties either. This fact is true for all linear random number generators, not just LCGs.

There is a type of linear generator available that preserves the good properties of the LCG while improving upon some of its disadvantages. We increase the order of the linear recursion and obtain the *multiple recursive congruential generator* ("MRG").

Example 1. Let $m\geq 2$ be the modulus, $k\geq 1$ be the order of the recursion and choose $a_0, a_1,\ldots, a_{k-1}$ in $Z_m:=\{0,1,\ldots, m-1\}$ with gcd(a_0, m)=1. Then

$$y_{n+k} \equiv \sum_{j=0}^{k-1} a_j y_{n+j} \pmod m, \quad n \geq 0$$

defines an MRG with initial values $y_0,\ldots, y_{k-1}$. We use the normalization $x_n:=y_n/m$ to produce random numbers x_n in the unit interval [0,1[. The maximum period of this generator is m^k-1, see [55,35]. The paper [35] contains tested parameters for MRGs with moduli m up to 2^{63} and code for one MRG in C. We will denote this MRG by "MRG1" and exhibit test results for it in Section 7.

One general problem with linear methods is the fact that correlations between random numbers separated by lags may be rather strong. Even in certain variants of the MRG, like the AWC and SWB

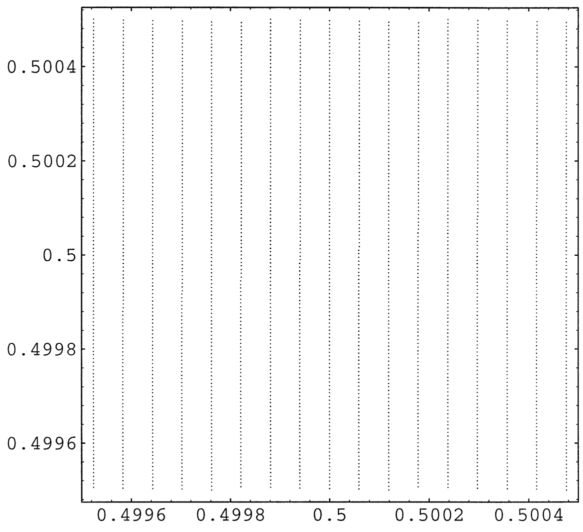

Fig. 3. Minimal Standard: LCG($2^{31}-1$,16807,0,1).

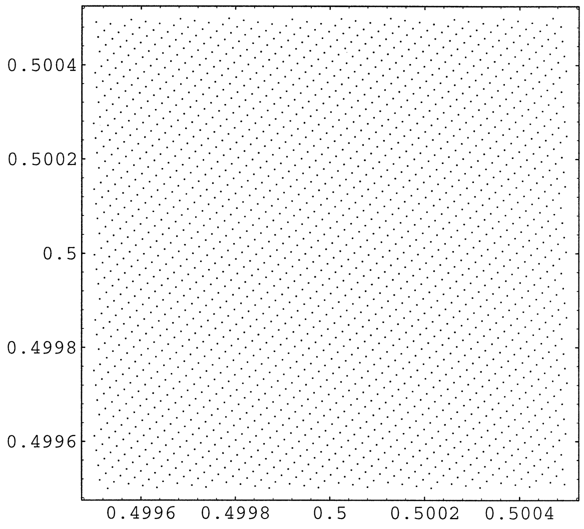

Fig. 4. SIMSCRIPT: LCG($2^{31}-1$,630360016,0,1).

generators of [46], this may lead to a very unfavorable performance in simulations, see [29,34,55] for further information. One solution to this problem is to combine generators. In the simplest version of this technique, we combine two generators by adding their output sequences $(x_n^{(1)})_{n\geq 0}$ and $(x_n^{(2)})_{n\geq 0}$ to obtain a new sequence $(x_n)_{n\geq 0}$,

$$x_n := x_n^{(1)} + x_n^{(2)} \pmod 1, \quad n \geq 0,$$

If the two generators are chosen properly, then the period of the sequence $(x_n)_{n\geq 0}$ will be the product of the periods of the components. Combining generators without theoretical support may lead to disastrous generators. In the case of the LCG and MRG, the theory is well-known, see [32,34]

Example 2. In [32], combined MRGs (“cMRG”) were introduced and thoroughly analyzed. Further, a particular cMRG was assessed by the spectral test up to dimension d=20 and its implementation in *C* was given. In Section 7, we will refer to this particular cMRG as “cMRG1”.

Tausworthe generators can have unacceptably bad empirical performance. For this reason, in [33], combined Tausworthe generators (“cTG”) were introduced to improve on the properties of single Tausworthe generators.

Example 3. In [33], an implementation in *C* of a cTG is given that has a period length of order 2^{88}. Further, the equi-distribution properties of this type of generator are analyzed. In Section 7, we will present test results for this generator, which is denoted by “cTG1”.

It is well-known that generalized feedback shift-register generators (“GFSR”) are fast, although a little bit tricky to initialize. Recently, a very interesting variant of this linear method has been presented in [49,50], the twisted GFSR (“tGFSR”). This generator produces a sequence $(x_n)_{n\geq 0}$ of w-bit integers by the rule

$$x_{n+p} = x_{n+q} \oplus x_n A, \quad n \geq 0,$$

where (w, p, q, A) are the parameters of the tGFSR and A is a $w\times w$ matrix with binary entries. This generator is fast and reliable if the parameters are chosen properly.

Example 4. The tGFSR "TT800" presented in Mat[50]94a has a period length of 2^{800} and strong theoretical support. We will exhibit convincing empirical evidence in Section 7.

Inversive generators were constructed to overcome one property of linear generators that may turn into a deficiency (depending on the simulation problem), the lattice structure of d-tuples of consecutive random numbers. There are several variants of inversive generators, inversive congruential generators ("ICG"), explicit-inversive generators ("EICG"), digital inversive congruential generators ("dICG"), and combinations of ICGs and EICGs. Inversion certainly slows down the generation of random numbers. Compared to LCGs of the same size, inversive generators are three to ten times slower, depending on the processor's architecture (see [43]).

The importance of inversive random number generators stems from the fact that their intrinsic structure and correlation behavior are strongly different from linear generators. Hence, they are very useful in practice for verifying simulation results. We refer the reader to [17] for a concise survey of the ICG and EICG, in comparison to LCGs. A comprehensive discussion of all available nonlinear methods is contained in [55]. The implementation of inversive generators is discussed in [43]. This generic implementation in *C* is also available from the server **http://random.mat.sbg.ac.at/**. In the case of the ICG and EICG, composite moduli lead to less convincing generators than prime moduli.

At the present state of the art, the dICG is slower than the ICG. From a disappointing speed factor of about 150 in the first implementation (see [[8] page 72]) this disadvantage has now been reduced to a factor less than 8, see [58].

For a given prime number p, and for $c \in Z_p$, let $\bar{c} := 0$ if $c=0$ and $\bar{c} := c^{-1}$ if $c \neq 0$. In other words, $\bar{c}$ equals the number c^{p-2} modulo p.

Example 5a. Inversive congruential generators ("ICG") were introduced in Eic[9]86a. We have to choose the modulus p, a multiplier a, an additive term b, and an initial value y_0. Then the congruence

$$y_{n+1} \equiv a\bar{y}_n + b (\operatorname{mod} p), \quad n \geq 0, \tag{1}$$

defines an ICG. We denote this generator by ICG(p, a, b, y_0). It produces a sequence $(y_n)_{n \geq 0}$ in the set $Z=\{0, 1, ..., p-1\}$. Pseudorandom numbers x_n in [0,1[are obtained by the normalization $x_n := y_n/p$.

A prominent feature of the ICG with prime modulus is the absence of any lattice structure, in sharp contrast to linear generators. In the following scatter plot, all possible points (x_{2n}, x_{2n+1}), $n \geq 0$, in a region near the point (0.5, 0.5) are shown in Fig. 5.

Example 5b. Explicit inversive congruential generators ("EICG") are due to [11]. The EICG is easier to handle in practice, for example when producing uncorrelated substreams. The cost is a slightly smaller maximum usable sample size, as empirical tests have shown (see [63,42]). We choose a prime number p, a multiplier $a \in Z_p$, $a \neq 0$, an additive term $b \in Z_p$, and an initial value n_0 in Z_p. Then

$$y_n \equiv \overline{a(n+n_0)+b} \quad (\operatorname{mod} p), \quad n \geq 0,$$

defines a sequence of pseudorandom numbers in $\{0,1,\ldots, p-1\}$. As before, we put $x_n := y_n/p$, $n \geq 0$, to obtain pseudorandom numbers in [0,1[. We shall denote this generator by EICG(p, a, b, n_0). In the definition of EICG(p, a, b, n_0), the additive term b is superfluous and can be omitted, see [18,43].

It is easy to create ICG and EICG on demand. The choice of parameters for the EICG is simple. In case of the ICG, we may use a "mother–child" principle that yields many ICGs from one "mother" ICG (see [17]).

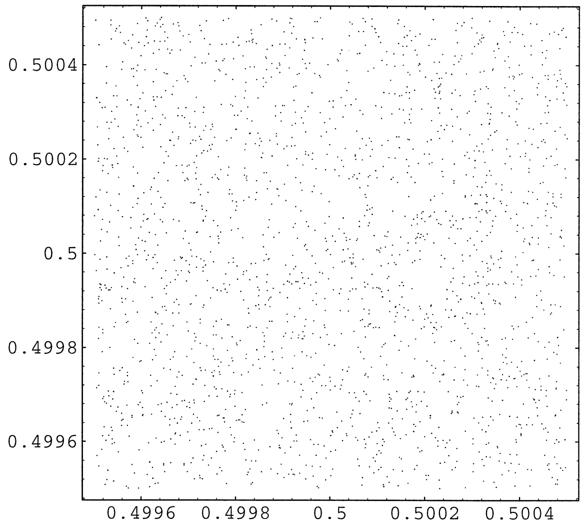

Fig. 5. All points of ICG($2^{31}-1$,1288490188,1,0).

The "compound approach" presented in [10,12] allows to combine ICG and EICG, provided they have full period. This method has important advantages: we may obtain very long periods easily, modular operations may be carried out with relatively small moduli, increasing the effectiveness of our computations, and the good correlation structure of the ICG and EICG is preserved. The price to pay is a significant loss of speed that makes combined inversive generators considerably slower than linear generators of comparable period length.

4. Theoretical support

Theoretical support for random number generators is still widely ignored by practitioners. The three main questions here are period length, the intrinsic structure of the random numbers and -vectors produced by a generator, and correlation analysis.

It is clear that the period length of a generator will put a limit to the usable sample size. Random number generation is equivalent to drawing without replacement, see [31]. Hence, the sample size should be much smaller than the period length of the generator. In the case of linear methods, the square root of the period length seems to be a prudent upper bound for the usable sample size. This recommendation is based on empirical experience, there is no theoretical analysis available (see [44] for a short discussion). We refer to Section 7 for examples that show how different types of random number generators behave quite differently when the sample size is increased.

In the case of good random number generators, it is possible to provide conditions for the parameters of the generator to obtain maximum period length, see [55]. Further, it is important to have algorithms at hand to compute such parameters. The case of the ICG is a good illustration for this requirement, see [17,55].

Intrinsic structures of random number generators like grid structures and related results like estimates of the number of points on hyperplanes are important to be known. For example, if one is aware of the grid structure of LCGs, then it will come as no surprise that this type of generator has difficulties with certain simulations. An instructive example is the nearest pair test, see [9] and [28,37].

The most difficult and most important part of the theoretical assessment of random number generators is correlation analysis. More than twenty years of experience have shown that certain figures of merit for random number generators allow very reliable predictions of the performance of the samples produced with a generator in empirical tests. The latter are nothing less than prototypes of simulation problems. Hence, the importance of theoretical correlation analysis for numerical practice is beyond question. It should be stated clearly that none of these figures of merit can give us guarantees for the performance of the generator in our simulation. At present, there is no firm mathematical link between any figure of merit for random number generators and the empirical performance of samples. Nevertheless, and this fact is truly remarkable, the quality of prediction is excellent.

The basic concept to analyze correlations between random numbers is the following. Suppose we are given random numbers $x_0, x_1, \ldots$ in the unit interval [0,1[. To check for correlations between consecutive numbers, we construct either overlapping d-tuples $\boldsymbol{x}_n := (x_n, x_{n+1}, \ldots, x_{n+d-1})$ or non-overlapping d-tuples $\boldsymbol{x}_n := (x_{nd}, x_{nd+1}, \ldots, x_{nd+d-1})$ and assess the empirical distribution of finite sequences $\omega = (\boldsymbol{x}_n)_{n=0}^{N-1}$ in the d-dimensional unit cube $[0,1[^d$. The task is to measure how "well" ω is uniformly distributed. Strong correlations between consecutive random numbers will lead to significant deviations of the empirical distribution function of ω from uniform distribution, in some dimensions d. It is clear that the restricted type of d-tuples that is considered here cannot ensure against long-range correlations among the numbers x_n themselves. For this topic, we refer the reader to [5]. In the case of the EICG of [11], more general types of d-tuples have been considered (see also [54,55]).

Interestingly, it has turned out that the behavior of full-period sequences ω with respect to theoretical figures of merit allows very reliable predictions of the performance of the random numbers x_n themselves in empirical tests. If the full-period point set ω has a good empirical distribution with respect to certain figures of merit in various dimensions d, then good empirical performance of the samples is highly probable. Practical evidence is that many target distributions will be simulated very well, see, for example, the empirical results in [15,28,42,23]. This relation between properties of full-period sequences in higher dimensions and the behavior of comparatively small-samples in low dimensions has not yet been put into rigorous mathematical form.

There are two schools of thought. The approach of the first school (see [30]) is to optimize the parameters of a given type of generator such that the empirical distribution function of the point sets ω in $[0,1[^d$ approximates uniform distribution as closely as possible, leading to a so-called "super-uniform" distribution. This is done in as many dimensions d as is feasible in practice. The figure of merit that is used for this task is the *spectral test*, due to [3]. The spectral test has an important geometrical interpretation as the maximum distance between successive parallel hyperplanes covering all possible points $\boldsymbol{x}_n$ that the generator can produce. This interpretation leads to efficient algorithms to compute the value of the spectral test. We refer the reader to [26,57,16,14,30,31,34,61] for details. The generator RANDU of Figs. 1 and 2 is not bad with respect to the spectral test in dimension d=2, but the value of the spectral test in dimension 3 is extremely small, thereby reflecting the catastrophic lattice structure in this dimension. This example explains why we have to consider the spectral test for a whole range of dimensions and compute its value for each of them.

The approach of the second school (see [55]) is to construct generators where the empirical distribution function does not approximate uniform distribution "too well". The maximum distance between the empirical distribution function and uniform distribution is preferred to be of order $1/\sqrt{N}$, where N denotes the number of points $\boldsymbol{x}_n$ we consider. This is, roughly speaking, and thinking of the law of the iterated logarithm "LIL" for discrepancy, the order of this quantity in the case of realizations of i.i.d. random variables on $[0,1[^d$ (see [25,55]). We will call this kind of equidistribution "LIL-uniformity". The figure of merit that is used here is the two-sided Kolmogoroff–Smirnov test statistic, also known as discrepancy in number theory. In terms of discrepancy, super-uniformity has an order of magnitude $\mathcal{O}((\log N)^d/N)$. Niederreiter has developed a powerful number-theoretic method to estimate discrepancy by exponential sums, see [51,53,55]. This method has allowed to assess this figure of merit for most types of random number generators.

Both the spectral test and discrepancy have their advantages and shortcomings. The advantage of the spectral test is that it is readily computable even in higher dimensions (i.e. above d=20) under the condition that points $\boldsymbol{x}_n$ have lattice structure, see [39,26,57,61]. This will only be the case for full-period sequences ω and for certain types of generators, mostly linear ones. Discrepancy is not limited to point sets with lattice structure. It is well-defined for every sample ω. Unfortunately, it is not possible to compute its value in practice, due to a complexity of order $\mathcal{O}(N^d)$, where N denotes the number of points and d the dimension. There are only upper and lower bounds available, due to Niederreiter's advanced method. For both figures of merit, their distribution in dimensions $d\geq 2$ is not known. Therefore, we cannot design an empirical test for random number generators from this quantities. In dimension one, the commulative distribution function ("c.d.f.") of discrepancy is known. We refer the reader to [26,53,55] for details. Recently, a probabilistic algorithm has been presented by [64].

There is a new addition to this list of figures of merit, the *weighted spectral test*. It is due to [18]. This figure of merit is derived from the original concept of the spectral test. It is related to discrepancy, can be estimated as the latter, it does not require lattice structure for the point sets ω, and it takes $\mathcal{O}(d\ N^2)$ steps to compute it in any dimension d, see [20,19]. The weighted spectral test may be interpreted as a mean square integration error, see [22]. Results on its distribution are already available, see [41,22].

Beyer quotients are another figure of merit to assess lattices. Unfortunately, this quantity is known to be defined properly only in dimensions up to 6. Beyond this dimension, the Minkowski-reduced lattice bases involved need not be unique any more and their Beyer quotients might be different (see [[40], 4.3.1]). For this reason, any results on bad Beyer quotients in dimensions higher than 6 are without mathematical justification at the present state of the art. A wrong basis might have been used.

5. Empirical evidence

Theoretical support for random number generators is not enough. Empirical evidence is indispensable. Every empirical test is a simulation. If selected with care, then it will cover a whole class of simulation problems. As we have indicated before, nothing can be deduced from the results of an empirical test if the practitioner uses completely different parameters in his own simulation problem.

It is relatively easy to design an empirical ("statistical") test for random numbers. Every function of a finite number of $U(0,1)$-distributed random variables whose distribution is known and which can be computed efficiently will serve for this purpose. What really matters here is to design tests that constitute prototypes of simulation problems and measure different properties of random numbers. Hence, every test in our battery should represent a whole class of empirical tests. No serious effort has yet been undertaken to classify the many empirical tests available according to this principle.

There are well-established batteries of empirical tests, see [26,45,28]. Marsaglia's DIEHARD battery is available on CD-ROM and from the Web-server **http://stat.fsu.edu/~geo/diehard.html**.

Several test statistics have been found to be rather discriminating between random number generators. In the class of bit-oriented tests that count the number of appearances of certain blocks of bits, Marsaglia's M-tuple test is outstanding, see [45,63,42,24]. Other reliable tests are the run test (see [26]) and a geometric quantity, the nearest-pair test (see [28,57]). Recently, a discrete version of an entropy test has been presented in [36]. It is not yet clear if this interesting test is really a new prototype not covered by the M-tuple test as employed in [63]. This example shows that, while it is relatively easy to design a new empirical test for random number generators, it is a nontrivial task to show that the new quantity is a meaningful addition to the established batteries of tests and strongly different from known test statistics.

On the basis of our practical experience we recommend the following approach to test design. Suppose we use a random variable Y with known c.d.f.F_Y. With the help of a random number generator, we produce K realizations $y_1,\ldots,y_K$ of this random variable. In the second step, we compare the empirical distribution function $\hat{F}_Y$ of the samples to the target distribution F_Y by some goodness-of-fit test, like the Kolmogoroff–Smirnov (KS) statistic. This procedure is called a two-level test, see [28,30]. Two-level test designs are a good compromise between speed and power of a test, see [28] for details.

If we want to test a random number generator without a particular application in mind, then it makes more sense to choose a smaller number of strongly different test statistics and to vary the parameters of the tests (like the sample size or the dimension) within large intervals. In our opinion, it is less relevant for practice to run an enormous battery of tests without any idea if all these tests really measure different properties of the generator. Further, if we do not vary the parameters enough and work, for example, with fixed sample sizes in our tests then our chances to meet the user's needs are small.

6. Practical aspects

Several aspects of a random number generator are of practical importance. For implementation, we need *tables of parameters* for good random number generators. Without *portable implementations* a generator will not be useful to the simulation community. Power users need *large samples*. For certain generators, in particular linear types, the limit for the usable sample size is close to $\sqrt{P}$, P the period of the generator, in many empirical tests. On 32-bit machines, most software packages work with LCGs of period length below 2^{32}. Hence, the maximum usable sample size is about 2^{15}, which is much too small for demanding simulations.

Parallel simulation creates additional problems (see [1]). Even reliable generators are unsafe when submitted to parallelization techniques. Basically, there are the following methods to generate random numbers on parallel processors. We may assign (i) L different generators to L different processors, or (ii) L different substreams of one large-period generator to the L processors. Technique (ii) has two variations. Either we use (a) a "leap-frog" method where we assign the substream $(x_{nL+j})_{n\geq 0}$ to processor j, $0\leq j<L$, or (b) we assign a whole segment $(x_n)_{n\geq n_j}$ to processor j, where $n_1,\ldots,n_L$ is an appropriate set of initial values that assures disjointness of the substreams. Technique (b) is called "splitting" of a random number generator.

Approach (i) cannot be recommended in general. There are no results on correlations between different random number generators, with one notable exception. For the EICG the correlation

Table 1
Spectral test for dimensions 2 to 8

d=2	3	4	5	6	7	8
0.2562	0.0600	0.0114	0.0462	0.1275	0.2031	0.2077

behavior of parallel generators has been analyzed. It was found to be remarkably good, see [52,54].

Approach (ii) is also unsafe territory. Linear methods like the LCG or MRG may occasionally (and unexpectedly) produce terrible sub-sequences with the leap-frog technique. We will illustrate this point with an example from [13]. Even a good generator like LCG(2^{48}, 55151000561141, 0) (see [14]) produces a leap-frog subsequence that performs even worse than RANDU in the spectral test. If we happen to assign the leap-frog subsequence $(x_{23n})_{n\geq 0}$ to one processor, then the values of the spectral test show that this was an unfortunate decision which we might regret, see Table 1.

Splitting is not safe either. So-called *long-range* correlations are lurking in the shadows (see [4,6]). Again, the EICG recommends itself for empirical testing, due to strong theoretical support with respect to splitting, see [11,52,54].

Available parallel random number generator libraries are based on linear algorithms. We refer the reader to [48,38,47,56,21]. These libraries should be used with the above warnings in mind.

7. Examples revisited

In the preceding sections, we have discussed several aspects of a good random number generator. We will now revisit the generators we have presented in Section 3.

Examples 1,2,3, and 4 have been constructed such that super-uniformity is achieved in as many dimensions as possible. For MRG1 and cMRG1 (see Examples 1 and 2) the parameters have been chosen with the spectral test, see [35,32]. In Example 3, theoretical analysis yielded conditions for optimal equi-distribution properties. These conditions allowed to perform exhaustive searches for optimal parameters, see [33]. A similar approach based on deep theoretical analysis was used in [50] to find good tGFSR like TT800.

Inversive generators are much less sensible to the choice of parameters. They yield LIL-uniformity once maximum period is assured by the parameters.

We will now report on the performance of our examples in a stringent test, Marsaglia's M-tuple test (see [45]). The test design and the graphic presentation of the results have been developed in [63]. We refer to this thesis for details of our setup.

From every random numbers x_0,x_1...$\in$[0,1[, we take the first consecutive r blocks of 4 bits in its binary representation, r=6,8. This procedure gives a sequence of random numbers $(y_n)_{n\geq 0}$ in the range {0, ..., 15}. We then consider the overlapping d-tuples $(y_n, y_{n+1}, \ldots, y_{n+d-1})$ and apply the overlapping M-tuple test, where d=4,5. For a given sample size N, we compute 32 values of the, theoretically equi-distributed, upper tail probability of the M-tuple test. In the following figures, the sample size ranges between 2^{18} and 2^{26}. The sample size is given in a logarithmic scale.

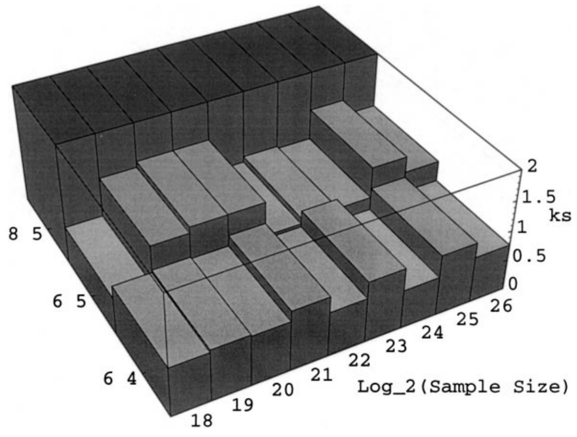

Fig. 6. MRG1: KS values.

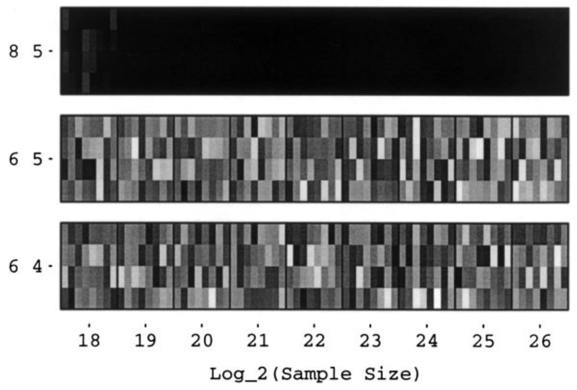

Fig. 7. MRG1: Upper tail probabilities.

In Fig. 6, we show the result of a two-sided KS test applied to these 32 values for MRG1. Values of the KS test statistic greater than the critical value 1.59 that corresponds to the significance level of 1% are shown in dark grey and indicate that the generator has failed the test.

In Fig. 7, we plot the 32 equi-distributed values of this test statistic. The resulting patterns should be irregular. If, for a given sample size N, the corresponding box is either totally white or black, the generator has failed. White indicates too good approximation (which is a result of super-uniformity), black signals too large deviation from the expected values. We observe that the MRG performs well for the first 6 blocks of digits of length 4 in dimensions 4 and 5, but it fails to simulate the theoretical distribution if we consider 8 blocks of 4 bits in dimension 5 due to the fact that it is a 31-bit generation. The combined generator cMRG1 (see Example 2 in Section 3) yields similar results and is therefore omitted.

The cTG of Example 3 performs considerable better, as the Figs. 8 and 9 shows. cTG1 has no problems with 32 bits. The tGFSR "TT800" (Figs. 10 and 11 of example 4) gives a flawless performance. It is the only generator here without rejections.

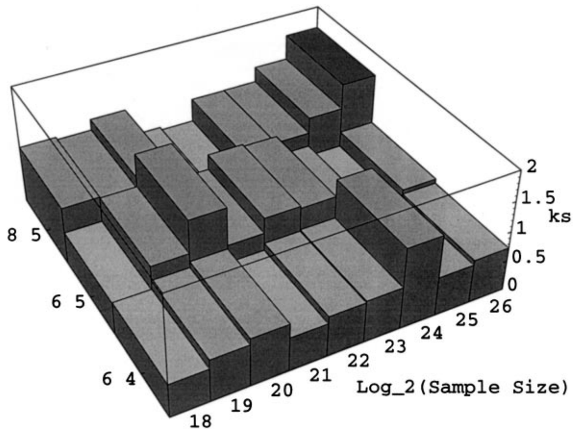

Fig. 8. cTG1: KS values.

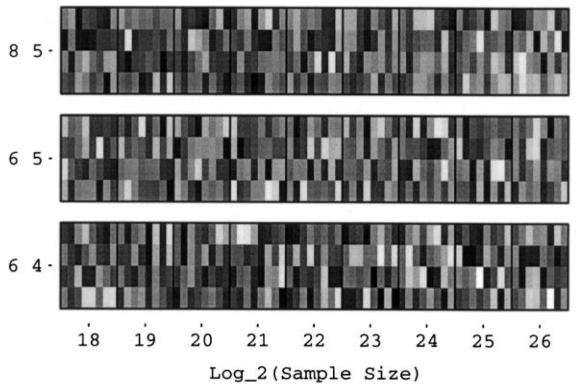

Fig. 9. cTG1: Upper tail probabilities.

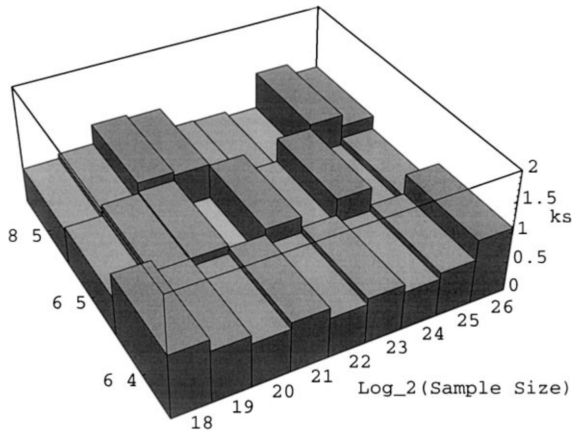

Fig. 10. TT880: KS values.

In comparison to these long-period generators, we see that only a compound ICG of period length close to 2^{32} is able to keep up with the large generators above. cICG1 (Figs. 16 and 17) combines ICG(1031,55,1,0), ICG(1033,103,1,0), and ICG(2027,66,1,0). An ICG with period length $2^{31}-1$ like ICG1= ICG($2^{31}-1$,1288490188,1,0) finally becomes ''overloaded'' (Figs. 12 and 13). A LCG of the same period length will perform poorly, as our example shows (Figs. 14 and 15).

8. RNG survival kit

The following selection of papers and links allows an easy orientation in the field of random number generation.

At starting points, we recommend [30,34], which cover a broad range of aspects. For readers that are interested in the mathematical background, [55] contains a wealth of comments and references. There

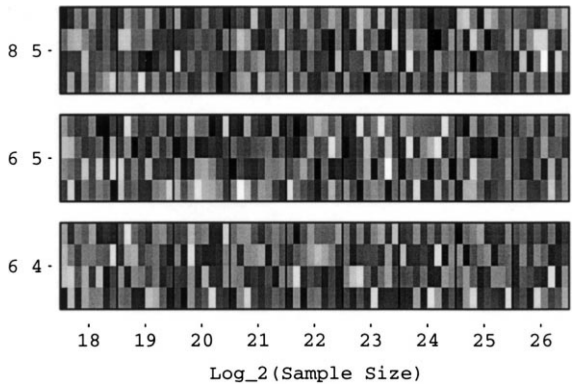

Fig. 11. TT800: Upper tail probabilities.

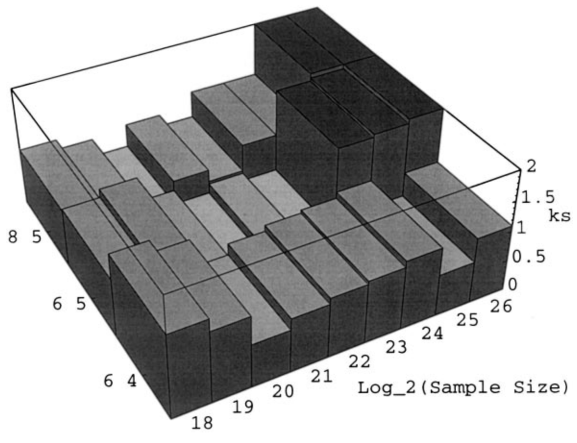

Fig. 12. cICG1: KS values.

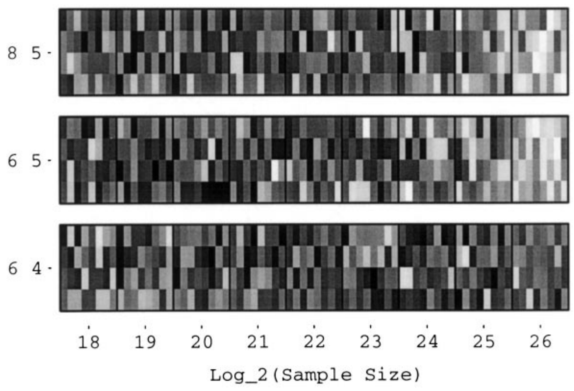

Fig. 13. cICG1: Upper tail probabilities.

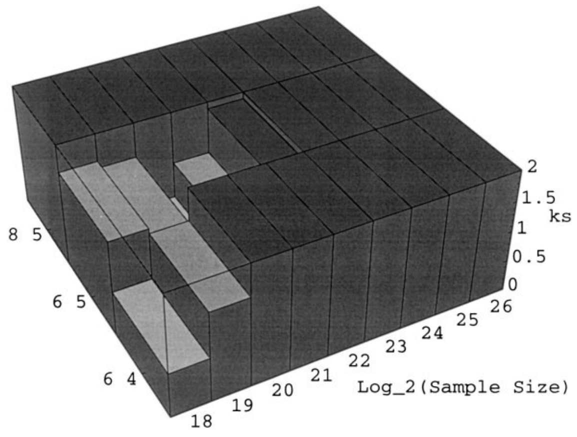

Fig. 14. IGG1: KS values.

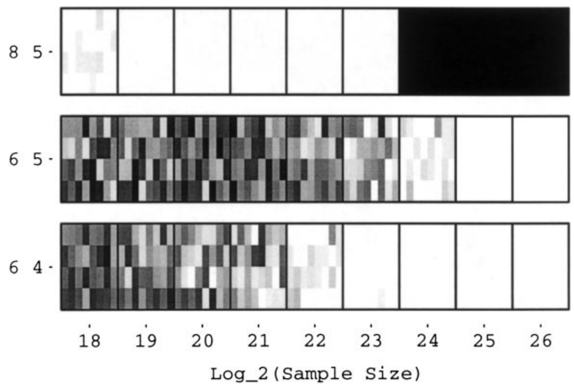

Fig. 15. ICG1: Upper tail probabilities.

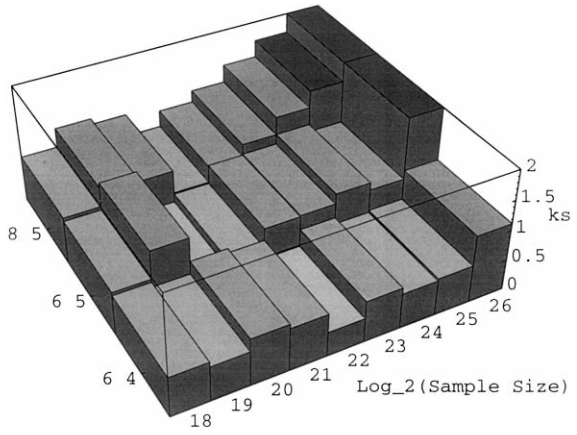

Fig. 16. ANSI-*C*: KS values.

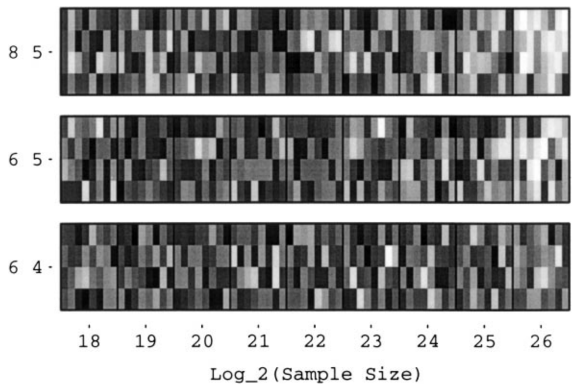

Fig. 17. ANSI-C: Upper tail probabilities.

are two monographs [53,61] in this field for further reading. [26] is considered to be the ‘‘bible’’ of random number generation.

Numerous links to information and software can be obtained from the Web site **http://random.mat.sbg.ac.at/**

9. RNG checklist

Theoretical support

Period length	conditions
	algorithms for parameters
Structural properties	intrinsic structures
	points on hyperplanes
	equidistribution prop.

Correlation analysis	for particular parameters
	for particular initializations
	for parts of the period
	for subsequences
	for combinations of RNG's

Empirical evidence

- Variable sample size
- Two-or higher level tests

bit-oriented tests
tests for correlations
geometric test quantities
complexity
transformation methods: sensitivity

Practical aspects

Tables of parameters available
Portable implementations available
Parallelization techniques apply
Large samples available

10. Summary

Random number generators are like antibiotics. Every type of generator has its unwanted side-effects. There are no safe generators. Good random number generators are characterized by theoretical support, convincing empirical evidence, and positive practical aspects. They will produce correct results in many, though not all, simulations.

Open questions in this field concern reliable parallelization, the creation of good generators on demand, the sensitivity of transformation methods (to obtain nonuniform random numbers) to defects of the uniform random number generators, the classification of empirical tests, and the mathematical foundation of forecasting the empirical performance by theoretical figures of merit.

There are three rules for numerical practice that are worth to keep in mind.

1. Do not trust simulation results produced by only one (type of) generator, check the results with widely different generators before taking them seriously.
2. Do not combine, vectorize, or parallelize random number generators without theoretical and empirical support.
3. Get to know the properties of your random number generators. (We have supplied pointers and a checklist for this task)

Nowadays, the tool-box of stochastic simulation contains numerous reliable random number generators. It is up to the user to make the best out of them.

Acknowledgements

I would like to thank my research assistants Stefan Wegenkittl, who has carried out the necessary computations for the figures in Section 7, and Karl Entacher, who has calculated Table 1.This work has been supported by the Austrian Science Foundation, project P11143/MAT.

References

[1] S.L. Anderson, Random number generation on vector supercomputers and other advanced architectures, SIAM Review 32 (1990) 221–251.
[2] A. Compagner, Operational conditions for random-number generation, Phys. Review E 52 (1995) 5634–5645.
[3] R.R. Coveyou, R.D. MacPherson, Fourier analysis of uniform random number generators, J. Assoc. Comput. Mach. 14 (1967) 100–119.
[4] A. De Matteis, J. Eichenauer-Herrmann, H. Grothe, Computation of critical distances within multiplicative congruential pseudorandom number sequences, J. Comp. Appl. Math. 39 (1992) 49–55.
[5] A. De Matteis, S. Pagnutti, Long-range correlations in linear and non-linear random number generators, Parallel Computing 14 (1990) 207–210.
[6] A. De Matteis, S. Pagnutti, Critical distances in pseudorandom sequences generated with composite moduli, Intern. J. Computer Math. 43 (1992) 189–196.
[7] L. Devroye, Non-Uniform Random Variate Generation, Springer, New York, 1986.
[8] C. Döll, Die digitale Inversionsmethode zur Erzeugung von Pseudozufallszahlen, Master's thesis, Fachbereich Mathematik, Technische Hochschule Darmstadt, 1996.
[9] J. Eichenauer, J. Lehn, A non-linear congruential pseudo random number generator, Statist. Papers 27 (1986) 315–326.
[10] J. Eichenauer-Herrmann, Explicit inversive congruential pseudorandom numbers: the compound approach, Computing 51 (1993) 175–182.
[11] J. Eichenauer-Herrmann, Statistical independence of a new class of inversive congruential pseudorandom numbers, Math. Comp. 60 (1993) 375–384.
[12] J. Eichenauer-Herrmann, Compound nonlinear congruential pseudorandom numbers, Mh. Math. 117 (1994) 213–222.
[13] K. Entacher, A collection of selected pseudorandom number generators with linear structures, Report, The pLab Group, Department of Mathematics, University of Salzburg, 1996.
[14] G.S. Fishman, Multiplicative congruential random number generators with modulus 2^{β}: an exhaustive analysis for β=32 and a partial analysis for β=48, Math. Comp. 54 (1990) 331–344.
[15] G.S. Fishman, L.R. Moore, A statistical evaluation of multiplicative congruential random number generators with modulus 2^{31}-1, J. Amer. Statist. Assoc. 77 (1982) 129–136.
[16] G.S. Fishman, L.R. Moore III, An exhaustive analysis of multiplicative congruential random number generators with modulus 2^{31}-1, SIAM J. Sci. Statist. Comput., 7, 24–45, 1986. Erratum, ibid, 7, 1058, 1986.
[17] P. Hellekalek, Inversive pseudorandom number generators: concepts, results, and links, in: C. Alexopoulos, K. Kang, W.R. Lilegdon, D. Goldsman (Eds.), Proceedings of the 1995 Winter Simulation Conference, 1995, pp. 255–262.
[18] P. Hellekalek, On correlation analysis of pseudorandom numbers, Proceedings of the Second International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Salzburg, July 9–12, 1996, Lecture Notes in Statistics, Springer, New York, 1997, to appear.
[19] P. Hellekalek, H. Leeb, Dyadic diaphony, Acta Arith., 1996, to appear.
[20] P. Hellekalek, H. Niederreiter, The weighted spectral test: diaphony, 1996, Submitted to ACM Trans. Modeling and Computer Simulation.
[21] M. Hennecke, Random number generators homepage, **http://www.uni-karlsruhe.de/~RNG/**.
[22] F. James, J. Hoogland, R. Kleiss, Multidimensional sampling for simulation and integration: measures, discrepancies, and quasi-random numbers, Preprint submitted to Computer Physics Communications, 1996.
[23] B. Johnson, Radix-*b* extensions to some common empirical tests for pseudo-random number generators, To appear in ACM Trans. Modeling and Computer Simulation, 1996.

[24] K. Kankaala, T. Ala-Nissila, I. Vattulainen, Bit-level correlations in some pseudorandom number generators, Phys. Rev. E 48 (1993) 4211–4214.
[25] J. Kiefer, On large deviations of the empiric d.f. of vector chance variables and a law of the iterated logarithm, Pacific J. Math. 11 (1961) 649–660.
[26] D.E. Knuth, The Art of Computer Programming, Vol. 2, Addison-Wesley, Reading, Mass., 2nd ed., 1981.
[27] J.C. Lagarias, Pseudorandom numbers, Statistical Science 8 (1993) 31–39.
[28] P. L'Ecuyer, Testing random number generators, in: J.J. Swain et al., (Ed.), Proc. 1992 Winter Simulation Conference (Arlington, Va., 1992), pp. 305–313, IEEE Press, Piscataway, NJ, 1992.
[29] P. L'Ecuyer, Bad lattice structures for vectors of non-successive values produced by some linear recurrences, 1994, to appear in ORSA J. on Computing.
[30] P. L'Ecuyer, Uniform random number generation, Ann. Oper. Res. 53 (1994) 77–120.
[31] P. L'Ecuyer, Random number generators, in: S. Gass, C. Harris (Eds.), Encyclopedia of Operations Research and Management Science, Kluwer Academic Publishers, 1995.
[32] P. L'Ecuyer, Combined multiple-recursive random number generators, To appear in Operations Res. 44, 1996.
[33] P. L'Ecuyer, Maximally equidistributed combined Tausworthe generators, Math. Comp. 65 (1996) 203–213.
[34] P. L'Ecuyer, Random number generation, In Jerry Banks (Ed.), Handbook on Simulation, Wiley, New York, 1997.
[35] P. L'Ecuyer, F. Blouin, R. Couture, A search for good multiple recursive random number generators, ACM Trans. Model. Comput. Simulation 3 (1993) 87–98.
[36] P. L'Ecuyer, A. Compagner, J.-F. Cordeau, Entropy tests for random number generators, Submitted to ACM Trans, Modeling and Computer Simulation, 1996.
[37] P. L'Ecuyer, J.-F. Cordeau, Close-point spatial tests for random number generators, draft version, 1996.
[38] P. L'Ecuyer, S. Coté, Implementing a random number package with splitting facilities, ACM Trans. Math. Software 17 (1991) 98–111.
[39] P. L'Ecuyer, R. Couture, An implementation of the lattice and spectral tests for multiple recursive linear random number generators, INFORMS J. Comput., 1996, To appear.
[40] H. Leeb, Random numbers for computer simulation. Master's thesis, Institut für Mathematik, Universität Salzburg, Austria, 1995, Available from **http://random.mat.sbg.ac.at/**.
[41] H. Leeb, A weak law for diaphony, Rist++13, Research Institute for Software Technology, University of Salzburg, 1996.
[42] H. Leeb, S. Wegenkittl, Inversive and linear congruential pseudorandom number generators in empirical tests, To appear in ACM Trans. Modeling and Computer Simulation, 1996.
[43] O. Lendl, Explicit inversive pseudorandom numbers. Master's thesis, Institut für Mathematik, Universität Salzburg, Austria, 1996, Available from **http://random.mat.sbg.ac.at/**.
[44] N.M. MacLaren, A limit on the usable length of a pseudorandom sequence, J. Statist. Comput. Simul. 42 (1992) 47–54.
[45] G. Marsaglia, A current view of random number generators, in: L. Brillard (Ed.), Computer Science and Statistics: The Interface, Amsterdam, Elsevier Science Publishers B.V. (North Holland), 1985, pp. 3–10.
[46] G. Marsaglia, A. Zaman, A new class of random number generators, Ann. Appl. Prob. 1 (1991) 462–480.
[47] M. Mascagni, M.L. Robinson, D.V. Pryor, S.A. Cuccaro, Parallel pseudorandom number generation using additive lagged-Fibonacci recursions, Technical report, Supercomputing Research Center, Institute for Defense Analyses, 1994.
[48] N. Masuda, F. Zimmermann, PRNGlib: a parallel random number generator library. Tachnical report, Swiss Center for Scientific Computing, 1996, Available from **http://www.cscs.ch /Official/Publications.html**.
[49] M. Matsumoto, Y. Kurita, Twisted GFSR generators. ACM Trans, Model. Comput. Simul. 2 (1992) 179–194.
[50] M. Matsumoto, Y. Kurita, Twisted GFSR generators II, ACM Trans. Model. Comput. Simul. 4 (1994) 254–266.
[51] H. Niederreiter, Quasi-Monte Carlo methods and pseudo-random numbers, Bull. Amer. Math. Soc. 84 (1978) 957–1041.
[52] H. Niederreiter, New methods for pseudorandom number and pseudorandom vector generation, in: J.J. Swain et al., (Ed.), Proc. 1992 Winter Simulation Conference (Arlington, Va., 1992), IEEE Press, Piscataway, NJ, 1992, pp. 264–269.
[53] H. Niederreiter, Random Number Generation and Quasi-Monte Carlo Methods, SIAM, Philadelphia, 1992.
[54] H. Niederreiter, On a new class of pseudorandom numbers for simulation methods, J. Comp. Appl. Math. 56 (1994) 159–167.
[55] H. Niederreiter, New developments in uniform pseudorandom number and vector generation, in: H. Niederreiter, P.J.-S. Shiue, (Eds.), Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, volume 106 of Lecture Notes in Statistics, Springer, New York, 1995, pp. 87–120.

[56] D.V. Pryor, S.A. Cuccaro, M. Mascagni, M.L. Robinson, Implementation and usage of a portable and reproducible parallel pseudorandom number generator, Technical report, Supercomputing Research Center, Institute for Defense Analyses, 1994.
[57] B.D. Ripley, Stochastic Simulation, John Wiley, New York, 1987.
[58] K. Schaber, Digital inversive congruential generators. Master's thesis, Institut für Mathematik, Universität Salzburg, Austria, 1997, Available from **http://random.mat.sbg.ac.at/**.
[59] E. Stadlober, R. Kremer, Sampling from discrete and continuous distributions with c-Rand, in: G. Pflug, U. Dieter (Eds.), Simulation and Optimization, volume 374 of Lecture Notes in Economics and Math. Systems, Springer, Berlin, 1992, pp. 154–162.
[60] E. Stadlober, F. Niederl, C-Rand: a package for generating nonuniform random variates, In Compstat '94, Software Descriptions, 1994, pp. 63–64.
[61] S. Tezuka, Uniform Random Numbers: Theory and Practice, Kluwer Academic Publisher, Norwell, Mass., 1995.
[62] I. Vattulainen, T. Ala-Nissila, K. Kankaala, Physical models as tests of randomness, Phys. Rev. E 52 (1995) 3205–3214.
[63] S. Wegenkittl, Empirical testing of pseudorandom number generators, Master's thesis, Institut für Mathematik, Universität Salzburg, Austria, 1995, Available from **http://random.mat.sbg.ac.at/**.
[64] P. Winker, K.-T. Fang, Application of threshold accepting to the evaluation of the discrepancy of a set of points, Research report, Universität Konstanz, 1995.