History of uniform random number generation

Pierre L'Ecuyer

HISTORY OF UNIFORM RANDOM NUMBER GENERATION

Pierre L'Ecuyer

DIRO, GERAD, and CIRRELT
Pavillon Aisenstadt, Université de Montreal
Montréal (Québec), CANADA
and Inria Rennes Bretagne–Atlantique, FRANCE

ABSTRACT

Random number generators were invented before there were symbols for writing numbers, and long before mechanical and electronic computers. All major civilizations through the ages found the urge to make random selections, for various reasons. Today, random number generators, particularly on computers, are an important (although often hidden) ingredient in human activity. In this article, we give a historical account on the design, implementation, and testing of uniform random number generators used for simulation.

1 INTRODUCTION

The Romans already had a simple method to generate (approximately) independent random bits. Flipping a coin to choose between two outcomes was then known as "navia aut caput", which means "boat or head" (their coins had a ship on one side and the emperor's head on the other). Dice were invented much earlier that that: some 5000-years ones have been found in Iraq and Iran. They were also popular in India and China at least 4000 years ago, and then around the Mediterranean Sea, in particular in Egypt (Carr 2017, Glimne 2017). These dice had from 1 to 6 dots on each side, just like today. They did not have written numbers because written numbers were not yet invented. That is, humans invented random number generators before symbols to represent numbers! Other types of devices to produce random outcomes have been found in practically every part of the world. Everybody needs random numbers. Interestingly, in most ancient civilizations, people did not believe in randomness; they believed that the dice (or other device) were telling them the decision of a god. In a sense, they "understood" that their random numbers were not truly random.

Some 127 years ago, Galton (1890) designed a method to sample (approximately) from a given probability distribution using dice. He used cubic dice (with six faces) but after throwing the dice he was picking up each die by hand and placing it aligned in front of him, eyes closed, and considered the orientation of the upper face. This gives 24 possible outcomes per die (almost 4.6 random bits).

Nowadays, throwing dice and typing the outcome for a computer would obviously be much too slow. A first idea is to recycle the numbers. Before the emergence of computers, statisticians who needed random numbers for random sampling and probabilistic experiments decided to construct and publish large tables of so-called *random sampling* numbers. Those were the days when one could publish a scientific paper or even a book filled up with only random digits! Tippett (1927) published the first table, with 41,600 "random" digits taken from a 1925 census report, apparently on a suggestion from Karl Pearson. Several other followed, mostly in the late 1930's and in the 1940's. For example, Fisher and Yates (1938) published a table constructed by picking digits from a large table of logarithms to 20 decimal places. Statistical testing soon revealed obvious problems with many of those tables; in particular, certain digits appeared too often and others not enough.

In a landmark contribution, Kendall and Babington-Smith (1938) studied the philosophical and practical issues arising in the construction of tables of random sampling numbers, and the distinction that should

be made between quality criteria for these finite tables and the notion of independent random digits in probability theory. In the framework of probability theory, every possible table (for a given size) has the same chance of occurring, so one could argue that no table is better than any other! To handle this apparent paradox, Kendall and Babington-Smith (1938) introduced a notion of *locally random* finite sequence, loosely defined to mean that each reasonably long subsequence should have the appearance of being random and pass a certain set of simple statistical tests of randomness. Note that truly random infinite sequences do not satisfy this criterion; they contain arbitrary long (rare) subsequences that only repeat the same digit, for example. Obviously, one can only select a limited number of tests among the astronomical number of possible tests that can be applied to a finite sequence of a given length. Kendall and Babington-Smith (1938) proposed four types of statistical tests for the digits; the first one tests only the uniformity of the digits and the other three also test independence: (1) a frequency test in which the observed frequency of each digit is compared to its expected (theoretical) frequency; (2) a serial test that measures the frequencies for pairs of successive digits; (3) a poker test that counts the frequencies of specific blocks of 5 digits; and (4) a gap test that observes the number of positions between successive appearances of any given digit and counts the frequency of each gap size. In each case, the observed frequencies are compared to the expected (theoretical) ones via a chi-square statistic.

They also constructed a table of 100,000 random sampling numbers using a cardboard disk that rotated at about 250 turns per minute, but with some randomness in speed. The disk was divided in 10 equal sectors numbered from 0 to 9, with a light that flashed at random times (about every 2 seconds on average) to show the number at a specific position. The "flashed" numbers were observed by an attentive human and recorded with a pencil. The experiment is described in great details in Kendall and Babington-Smith (1939). It was apparently the first time a "machine" was used to generate random numbers, but still the process was not completely automated. They submitted their digits to the four statistical tests they had defined, first by testing each of the 100 blocks of 1000 digits separately, then 20 blocks of 5000 digits, then four blocks of 25,000 digits. In the 4×124 resulting p-values, they found that less than 2% were outside the interval $(0.01, 0.99)$, so they concluded that their table passed the tests. They also found that the table of Tippett (1927) passed their tests. At about the same time, Kermack and Kendrick (1937a, 1937b) considered certain types of *run tests* and *gap tests* to detect periodic behavior. They constructed a table of "random" digits taken from a telephone directory, and the table passed their tests. Kendall and Babington-Smith (1938) also tried taking digits from a (different) telephone directory and found bias (the frequency of certain digits was much too small).

In the 1940's and 1950's, following the introduction of electronic computers, one could get a large deck of punched cards filled with random sampling digits taken from such a table or produced in another way. Those cards could be placed in the "data" section of a simulation program in FORTRAN, for example. I did this myself in 1969. In 1947, the RAND Corporation in Santa Monica undertook a project to construct a table of one million random decimal digits (Brown 1949). It was the first table constructed in a fully automated way, using a physical apparatus and a computer. They designed an electronic device in which pulses were emitted at random, about 10^5 pulses per second, and they counted the number of pulses emitted every second, modulo 32 (with a 5-bit counter). Twenty of the 32 possible values were mapped to decimal digits and the others were discarded. These digits were "punched" on 20,000 IBM cards, 50 per card, one digit per second. Statistical tests then revealed that odd digits were slightly (but significantly) more frequent than even digits on large portions of the sequence, due to imperfections of the physical device (the 32 possible values did not have exactly the same probability). To reduce the bias, it was decided to apply a *compound randomization* technique which provably reduces the bias by adding up modulo b two or more independent digits in base b (Horton and Smith III 1949). They simply transformed each digit by adding it modulo 10 to the corresponding digit of the previous card. This gave a new set of punched cards with the final random digits. Four types of empirical statistical tests were applied to the digits: the frequency, serial, and poker tests of Kendall and Babington-Smith (1938), and a test on run lengths of the same digit (Brown 1948). The tests found no problem with the final (transformed) digits. For a while,

these random digits were available only in a deck (or box) of punched cards. In 1955, they were printed in a book, 50 rows of 50 digits per page (RAND Corporation 1955). The book was reissued in paperback in 2001. It is a book with maximum possible suspense: when you read it sequentially for the first time, after every character, based on what you have read so far, you still have no clue of what comes after!

For physicists who used electronic computers for large-scale *Monte Carlo* simulations, reading the random numbers from punched cards or other external storage devices was too slow, and the size of main memory was much too limited to store large tables of random digits. Two types of solutions emerged to produce random numbers on the fly, in real time: (1) using a fast physical device that produces random noise and (2) using a purely deterministic algorithm to produce a sequence that *imitates* randomness. These are still the two main classes of approaches to generate random numbers on computers today. Physical devices are used (and required) in applications such as cryptography, lotteries, and gambling, for example. For simulation, algorithmic generators are more appropriate, mainly because they are much simpler and more convenient to use, and the sequences they produce are reproducible. The investigators in the RAND project already anticipated these algorithmic generators. Brown (1949) ended his report with the visionary comment: "... for the future ... it may not be asking too much to hope that ... some numerical process will permit us to produce our random numbers as we need them. The advantages of such a method are fairly obvious in large-scale computation where extensive tabling operations are relatively clumsy."

In Sections 2 and 3, we give a quick tour of the history of these two classes of methods (physical and algorithmic generators, respectively) from the 1940's until today. We examine early attempts for algorithmic generators and how the key quality requirements for those generators have emerged and evolved. In Section 4, we survey the development of empirical statistical tests for RNGs. Sections 5 and 6 covers the development of RNGs based on linear recurrences modulo a large integer m and modulo 2. Those are still the most popular methods for stochastic simulation and we explain why. Section 7 discusses various types of nonlinear generators. Section 8 covers the design of RNGs with multiple streams and substreams, and RNGs for parallel processors. Section 9 provides further general references.

Generating random variates from non-uniform distributions is done by applying appropriate transformations to uniform random numbers. This is covered in a separate article elsewhere in these proceedings and in Devroye (1986).

2 GENERATORS BASED ON PHYSICAL DEVICES

Flipping coins, throwing dice, picking balls from an urn, shuffling cards, etc., have been used for centuries to produce randomness, but with the advent of computers, faster methods were needed. Early electronic devices included counters of random events (like the one used for the RAND table) and periodic sampling of electric noise. These devices operate at a microscopic level. The counters count the number of impulses of some sort, e.g., electrical or radioactive, modulo some small integer (often 2 or a small power of 2), over a given time interval. The noise sampling devices sample a signal periodically and output one or more random bits that depend on the signal level, at each sampling time. In one of the earliest publications on this topic, Cobine and Curry (1947) proposed a system in which electric noise produced in a gas tube in a magnetic field was collected, amplified, and sampled to produce random bits.

A famous machine named ERNIE (Electronic Random Number Indicator Equipment) that could produce about 50 random digits per second was put in use in 1957 to determine winning numbers in the British Savings Bonds Lottery (Thomson 1959). There were actually several thousand winning numbers to generate at each draw. To produce noise, a high voltage was applied at each end of glass tubes filled with neon gas, to produce current inside the tube. Collisions between the electrons and the neon atoms make the current noisy, the noise is amplified and collected by some additional systems, and transposed into random digits. These digits passed all statistical tests that were applied. The original machine was used until 1972, then upgraded three times to smaller and faster models. ERNIE 4, the current model since 2004, extracts random bits from thermal noise in transistors.

Since around 1950, thousands of articles have proposed physical devices that produce random bits or random numbers and nearly 2000 patents have been issued for such devices. See Herrero-Collantes and Garcia-Escartin (2017) and Stipčević and Koç (2014) for reviews. The classical ones use various types of thermal and electric noise, photoelectric effect, the low digits of clock readings or disk access times, etc. Recently, there is an emphasis on devices based on quantum physics phenomena, such as light beam splitters (in which photons hit a semi-reflective surface and take any of two directions with probability close to 1/2), shot noise in electronic circuits, radioactive decay detected by a Geiger counter, etc.

The output of these devices is "random", but this does not suffice. If the device generates a sequence of bits, each bit should be 0 or 1 with equal probability (exactly) and all the bits should be independent. But for practical physical devices this is never exactly true. The bits are slightly biased and/or slightly correlated because of imperfections in the construction. But there are techniques to reduce both bias and correlation to levels that are practically undetectable, by combining the bits in some ways. The technique proposed by Horton and Smith III (1949) of adding modulo b sequences of digits in base b, used with $b = 10$ for the RAND table, is one of them and it provably reduces the bias. In base 2, it amounts to xoring two or more blocks of bits and retaining the result. This is commonly used in practice. The following technique proposed by von Neumann (1951) completely eliminates the bias when there is no correlation: regroup all the bits by pairs, discard all the pairs 00 and 11, and replace the pairs 10 by 0 and 01 by 1. There are extensions of this method that can eliminate both the bias and the correlation (Blum 1986, Chor and Goldreich 1988, Peres 1992).

Today, several physical devices available on the market incorporate these types of methods to generate random bits or random digits. A comparative list can be found on Wikipedia (2017). The fastest ones can produce up to about 3 Gbits per seconds. Those hardware generators are needed for cryptography, lotteries, and in gambling machines, where true randomness is essential. Often, these applications combine an algorithmic RNG with a small source of entropy to inject some true randomness. For example, one can have the deterministic RNG running non-stop in the background to produce an output stream from which a few bits or numbers are taken when needed (at random times) and the state of the RNG is modified frequently (e.g., every few seconds) with random bits coming from the entropy source. This approach is common in gambling machines.

Hardware RNGs are not appropriate for simulation for many reasons. It was recognized already in the early 1950's that physical devices are complicated to install and run, costly, slow, and much more importantly cannot reproduce the same sequence twice. Reproducibility is very important in simulation experiments, e.g., for comparing systems with common random numbers (Law and Kelton 1982, Bratley et al. 1983) and for program verification and debugging. This motivated the introduction of algorithmic RNGs.

3 ALGORITHMIC GENERATORS

A general framework for algorithmic RNGs was defined in L'Ecuyer (1990, 1994) as a structure $(\mathcal{S}, \mu, f, \mathcal{U}, g)$ where $\mathcal{S}$ is a finite set of *states* (the *state space*), μ is a probability distribution on $\mathcal{S}$ for the *initial state* (or *seed*) s_0, $f : \mathcal{S} \to \mathcal{S}$ is the *transition function*, $\mathcal{U}$ is the *output space*, usually the interval $[0,1)$ or a finite subset, and $g : \mathcal{S} \to \mathcal{U}$ is the *output function*. The state at step i is s_i and it evolves according to $s_i = f(s_{i-1})$, for $i \geq 1$. The *output* at step i is $u_i = g(s_i) \in \mathcal{U}$. These u_i are the *(pseudo)random numbers* produced by the RNG.

In 1946, von Neumann proposed what seems to be the first algorithmic RNG, known as the *middle-square method* (von Neumann 1951). The idea is to start with a $2a$-digit number s_0 in base b, where a is an integer (e.g., $2a = 10$ and $b = 10$). At each step, the number is squared to obtain a $4a$-digit number in base b, and the middle $2a$ digits of that number are retained and returned as random digits. Here, s_i is the $2a$-digit number at step i, the transformation f squares and takes the middle digits, g is the identity, and $\mathcal{S} = \mathcal{U}$ is the set of $2a$-digit numbers in base b. The method was in use for some time. However, it was found that depending on the initial number s_0, the sequence usually hits a fixed point s_i, where $f(s_i) = s_i$,

or a very short cycle, and this often occurs after a number of steps that is not very large (Metropolis 1956). For an illustration, suppose $b = 10$ and $2a = 2$, so we start with a two-digit number in base 10. Out of the 100 starting points, there are four fixed points, namely 0, 10, 50, 60, and one cycle of length 2, which is $24 \leftrightarrow 57$. We have $f(79) = 24$, so either 79, 24, or 57 leads to a cycle of length 2. All 97 other numbers lead to a cycle of length 1, at one of the four fixed points (Jansson 1966). Metropolis (1956) reports that in an experiment with $b = 2$ and $2a = 38$, with two starting points, about 500,000 numbers were generated before hitting 0 in one case, and about 750,000 numbers in the other case. Variants of the middle square method have been proposed as improvements, but they turned out to be not much better (Jansson 1966).

The idea of using successive decimals of π, e or other transcendental numbers to imitate a random sequence was also suggested very early. Metropolis et al. (1950) managed to compute 2,000 decimals of π and e, and found that this sequence passes elementary statistical tests. Such testing was extended to the first 10,000 decimals by Pathria (1962) and to 100,000 decimals by Esmenjaud-Bonnardel (1965), and no problem was found. Longer sequences of digits of π have been produced and tested afterward and many papers have discussed the idea. The world record in 2016 was 22,459,157,718,361 decimal digits of π, computed in about four months by Peter Trueb using an algorithm of Chudnovsky and Chudnovsky (1989), Bellard's formula, and the Y-Cruncher multi-threaded software (Yee 2017). However, a very significant drawback of using this sequence as an RNG is it would be much too slow, and would take too much space if we store the digits.

Moreover, to justify that the successive digits of π (or any other given real number) in a given base b can be taken as random, it would be good to know that this sequence of digits is *uniformly distributed in base b*, i.e., that each of the b possible digits appears with frequency $1/b$ (on average) in the infinite sequence. For π, empirical counting over several digits suggests that this might be true, but there is no known proof of it. Also, this uniform distribution is far from sufficient; we want for example that any digit appears with the same frequency after a 5, or after 25, or after 123, etc. This notion of *independence* of the successive digits is captured by the following stronger property introduced by Borel (1909): A sequence of digits in base b is *k-distributed* if any of the b^k possible blocks of k successive digits appears with the same frequency in the long run. We say it is *∞-distributed* if it is *k-distributed* for any $k \geq 1$. Borel (1909) called *normal to base b* a real number whose expansion in base b is ∞-distributed, and proved the remarkable fact that almost all real numbers (with respect to the uniform measure) are normal to any base. All normal numbers are irrational, and some can be constructed explicitly, but using them to define an RNG is not very efficient and practical, unfortunately.

For sequences of real numbers in the interval $[0,1)$, *uniformly distributed* (or *equidistributed*) means that for every interval $(a,b) \subseteq [0,1)$, the proportion of numbers from the infinite sequence that fall in (a,b) is $b-a$. This notion was defined and studied by Weyl (1916). There are many known constructions of such sequences. For example, for any irrational number $\alpha > 0$, the sequence $\{i\alpha \bmod 1, i \geq 1\}$ is uniformly distributed. This is easy to implement to a fixed number of bits of accuracy (e.g., 53 bits, for double precision floats). However, the successive numbers from an equidistributed sequence can be dependent. Again, a stronger and more relevant uniformity concept is that of an *∞-distributed* (or *completely uniformly distributed*) sequence, which means that for every $k > 0$, for every k-dimensional rectangular box B in $[0,1)^k$, the proportion of vectors of k successive numbers from the sequence that fall in B is equal to the volume of B. Korobov (1948) introduced this concept and studied it further in following papers. It was also studied by Franklin (1963) in the context of RNGs. Franklin (1965) showed in particular that for any transcendental number α, the sequence $\{u_i = \alpha^i \bmod 1, i \geq 1\}$ is ∞-distributed. Kuipers and Niederreiter (1974) provide an extensive coverage of these concepts.

Similar to ∞-distributed sequences of digits in base b, ∞-distributed sequences modulo 1 are not easy to implement for use as efficient RNGs. And even if they were, a long finite segment of such as sequence does not necessarily "look" random; see the detailed discussion in Section 3.5 of Knuth (1998). For example, an ∞-distributed sequence must contain an arbitrarily long segment in which all the values are less than 1/2. von Neumann (1951) already warned against programming an ergodic transformation that produces

a uniformly distributed sequence over $(0,1)$ and using it as an RNG, because when implemented with finite-precision arithmetic on a computer, the ergodicity disappears and the sequence is no longer uniformly distributed. It becomes periodic and may be very badly behaved.

The bottom line is that if we assume a fixed finite memory size for the state of the RNG, the sequence must be eventually periodic (i.e., it ends up in a cycle), so it is more relevant to study the behavior of the periodic sequence directly. This is what has been done in much of the research and development on RNGs since around 1950.

Forsythe (1951) describes one of the first methods to construct an RNG with known period. The idea, which Forsythe attributes to J. B. Rosser, is to combine several short periodic sequences of digits with relatively prime periods, by modular addition, to obtain a sequence whose period is the product of the individual periods. He examined and tested a specific binary sequence constructed with four sequences of periods 31, 33, 34, and 35, combined to obtain an overall period of 1,217,370. The resulting binary generator was used on the National Bureau of Standards Western Automatic Computer (SWAC). Interestingly, this combination principle to obtain long-period RNGs from shorter-period components has been rediscovered and reused in the RNG literature and is still used in some of the best RNGs currently available; see Wichmann and Hill (1982) and L'Ecuyer (1986, 1999a, 1999c, 2012), for example,

In a landmark contribution presented at a "Symposium on Large Scale Digital Calculating Machinery" at Harvard University in September 1949, Lehmer (1951) suggested the general idea of an algorithmic RNG with long and known period, explained that the middle-square method was unsatisfactory because it usually ends up in a very short cycle, and proposed the multiplicative linear congruential generator (LCG) as an RNG whose period can be very long and known under some conditions, thanks to number theory. The multiplicative LCG has transition function $x_i = a x_{i-1} \mod m$ and output $u_i = x_i/m$, for some integers $1 < a < m$. Lehmer gave two specific examples, one with $m = 10^8 + 1 = 17 \times 5,882,352$ and period $5,882,352$ for decimal arithmetic and another of $m = 2^{31} - 1$, $a = 23$, and period $2^{31} - 2$ for binary arithmetic. The first one was implemented on the IBM calculating punch 602A for statistical testing and could generate 1,250 random numbers (10,000 decimal digits) per hour on that machine. On the ENIAC (the supercomputer of the time), it was about 1800 times faster and could generate 5,000 decimal digits per second. In Lehmer's examples, the initial state x_0 was "chosen at random from a wastepaper basket of punched cards." His paper also contains an interesting discussion on the potential contribution of abstract mathematics to this new emerging field named "large-scaled computing", and vice-versa. His paper could have been subtitled "number theory to the rescue". Most modern long-period RNGs are still based on linear recurrences modulo some integer m and their period is computed by number theory arguments. We will continue their story in Sections 5 and 6.

With the LCGs, one was able to construct generators with known period and a certain guarantee of uniformity, because the LCG can be constructed to visit each integer between 0 and m exactly once in a cycle. But what about the uniformity of the k-tuples of successive values (which captures independence)? We will return to this in Section 5.

4 STATISTICAL TESTING

The number of different statistical tests that can be defined for RNGs is unlimited. The aim of a statistical test is to try to detect evidence against the null hypothesis $\mathscr{H}_0$ that the successive numbers produced by the RNG are independent with a given distribution, typically uniform over a given set; e.g., $\{0,1\}$, or the set of decimal digits, or the interval $(0,1)$. A statistical test is defined by any measurable function (a random variable) which maps any sequence of numbers from the given set to a real number X. To decide how well the realization x of X agrees with $\mathscr{H}_0$, one needs to know the distribution of X under $\mathscr{H}_0$, or at least a good approximation of this distribution. Then, given x and the distribution of X, one can compute the right and left p-values of the test, defined as $p^+ = \mathbb{P}[X \geq x]$ and $p^- = \mathbb{P}[X \leq x]$. When one of these p-values is very close to 0 (e.g., less than 10^{-6}), we have evidence against $\mathscr{H}_0$.

The development of appropriate statistical tests for RNGs had a somewhat bumpy history, in the sense that tests were often proposed based on incorrect distributions. We give some illustrations in what follows.

The early tests proposed by Kendall and Babington-Smith (1938) were defined for sequences of decimal digits. The frequency test, for example, counts the frequency of each possible digit in a sequence of length n, and compares these frequencies with their expectations of $n/10$ via a chi-square test statistic, which has approximately a chi-square distribution with 9 degrees of freedom if n is large enough. The two-dimensional serial test looks at all pairs of successive digits, overlapping or non-overlapping, and counts of how many times each of the 100 possibilities is observed. For a sequence of length $2n$, in the non-overlapping case, there are n pairs, and one can compare the observed counts with their expected values of $n/100$ via a chi-square statistic, which has approximately the chi-square distribution with 99 degrees of freedom. Kendall and Babington-Smith (1938) took n *overlapping* pairs in a sequence of length $n+1$ instead, i.e., they started a pair at each observation except the last one, and they still assumed a chi-square distribution, which is incorrect! This was pointed out by Good (1953), who studied the overlapping serial test in s dimensions for a sequence of length n, in two versions: (a) the $n-s+1$ s-tuples $(u_i,\ldots,u_{i+s-1})$, $i=0,\ldots,n-s$ are considered; (b) the sequence is put on a circle so one defines $u_i = u_{i-n}$ for $i \geq n$, and n tuples are considered, for $i=0,\ldots,n-1$. Good (1953) showed that in both cases, asymptotically when $n\to\infty$, the chi-square test statistic X_s^2 for the s-tuples does not have a chi-square distribution, but that the first and second differences with respect to s, namely $X_s^2 - X_{s-1}^2$ and $X_s^2 - 2X_{s-1}^2 + X_{s-2}^2$ (for $s>2$), have chi-square distributions with $d^{s-1}(d-1)$ and $d^{s-2}(d-1)^2$ degrees of freedom, respectively, where d is the number of distinct values to which the u_i's are mapped. In the serial test of Kendall and Babington-Smith (1938), $d=10$ and $s=2$, so the first difference $X_2^2 - X_1^2$ has a chi-square distribution with 90 degrees of freedom.

Kermack and Kendrick (1937a, 1937b) defined their *gap test* as follows. In a sequence of real numbers, we have a (local) maximum when the current number is larger than its two neighbors (or its single neighbor if we are at an extremity), and a minimum when it is smaller than its neighbor(s). They defined a *run up* as a finite sequence that starts at a minimum and ends at the next maximum (included), and a *gap* as the distance (number of observations) between a minimum and the next one. They obtained the probability distribution of the length of a run up (or run down) in an infinite random sequence. Their tests compute the average and standard deviation of the gap lengths over a fixed number of successive gaps, and compare them with the theoretical values.

Much earlier, Bienaymé (1875) stated (without proof) a central limit theorem for the total number of runs (up or down) in a sequence of n i.i.d. continuous random variables, when $n\to\infty$. It provides the asymptotic mean and variance as a function of n, and this can be used to define a runs test. Bienaymé (1875) observed that the behavior of runs and gaps does not depend on the distribution of the random variables in the sequence; it can be studied by simply assuming that we have a random permutation of n distinct numbers (selected in advance). Let $p_n(s)$ be the probability that a sequence of length n has exactly s runs. André (1884) derived a recurrence formula to compute these probabilities exactly:

$$(n+1)p_{n+1}(s) = sp_n(s) + 2p_n(s-1) + (n-s+1)p_n(s-2).$$

See Barton and Mallows (1965) for additional references and history.

A test based on just the total number of runs can detect some forms of correlation between successive values in the sequence, but more power against certain alternatives can be obtained by looking at the number of runs of each length and comparing these numbers with the theoretical expectations under $\mathscr{H}_0$, e.g., via a chi-square test. How can we obtain these expectations? Fisher (1926) gave the following formula for the probability $p(\ell)$ that a randomly selected run in an infinite sequence has length ℓ:

$$p(\ell) = \frac{3}{(\ell+1)!} - \frac{6}{(\ell+2)!} + \frac{3}{(\ell+3)!}.$$

This may suggest the following procedure. For a given (long) sequence of length n, compute the total number of runs, say S, and take $Sp(\ell)$ to estimate the expected number of runs of length ℓ. Levene and

Wolfowitz (1944) pointed out that this procedure, apparently used at the time, is erroneous because it neglects the (strong) dependence between S and the distribution of run lengths. An unbiased estimator is $Sp(\ell \mid S) \neq Sp(\ell)$, and the difference can be large. Levene and Wolfowitz (1944) derived exact expressions for the mean and the covariance matrix of the vector $\mathbf{r} = (r_1, \ldots, r_{d-1}, r_d^+)$ for a sequence of fixed length n, where r_j is the number of runs of length j, r_d^+ is the number of runs of length d or more, and d is a positive integer. The expression for the mean had already been given Kermack and Kendrick (1937a). (The run lengths are defined differently in these two papers; they are one less in Levene and Wolfowitz (1944) and in most later papers.) Wolfowitz (1944) also proved a central limit theorem for $\mathbf{r}$ when $n \to \infty$, which justifies a test based on a quadratic form in $\mathbf{r}$, whose asymptotic distribution is chi-square with known degrees of freedom. Knuth (1981) derives a slightly different run test based on the lengths of the runs up only (or on the runs down only). He argues that the development is easier this way, and that one can apply separate tests for the runs up and the runs down. In his third edition, Knuth (1998) adds an $\mathcal{O}(1/n)$ correction term to the covariance matrix of the chi-square test statistic. Alternatively, a much simpler version of the run test can be defined by skipping one value after each run. Then the successive runs up become independent and each run has length j with probability $1/j! - 1/(j+1)!$ (Knuth 1998, Exercice 3.3.2-14).

For sequences of independent uniform random digits in base b (b-ary sequences) another family of run tests can be defined in terms of successive digits that are the same. For $b = 2$, for example, we have a binary sequence and we may define a run as a string of 1's preceded and followed by 0's. We may look at the number of runs of each length, the length of the longest run, the total number of runs, etc., and define test statistics in terms of those. Sequences of real numbers can also be transformed into b-ary sequences to apply those tests; for example, for a sequence of real numbers in $(0,1)$, one may replace each number by 1 if it is above 1/2 and by 0 otherwise, and test the runs in the binary sequence. Extensive reviews of the theory underlying those tests can be found in Mood (1940), Fu and Koutras (1994), and Fu (1996).

References to other early test proposals from around 1940 can be found in Hull and Dobell (1962). Several more have been proposed afterward, many of them designed to be representative of specific applications or to detect known structures present in certain types of generators. Jansson (1966) and Knuth (1969) provide descriptions of the main tests recommended at the time; they include gap tests which count the number of steps between successive visits to a given interval for $U(0,1)$ random numbers, maximum-of-t tests which observe the maximum of t successive output values, poker tests, permutation tests, linear correlation tests, etc.

To test whether all digits appear sufficiently regularly in a b-ary sequence, Greenwood (1955) proposed the *coupon's collector* test, which counts the number X of values (digits) we need to generate to observe all the digits at least once. This is repeated n times and the distribution of the n realizations of X is compared with the theoretical distribution via a chi-square.

Gruenberger and Mark (1951) proposed a test based on the distribution of the square distance between two independent random points in the unit square. This square distance is computed for n independent pairs of points and the empirical distribution is compared with the theoretical one. Ripley and Silverman (1978) came up with a more general version in which n independent points are generated and one computes the distance between the two nearest points. L'Ecuyer et al. (2000) generalized it further in several directions: one can take other norms than the Euclidean one, the test can be based on the m nearest pairs of points, the unit square can be replaced by a higher-dimensional unit hypercube or unit torus, etc. Certain types of RNGs would never generate points that are very close to each other, and these authors showed that these types of *nearest pairs* tests can detect it systematically.

Several generalizations of the serial test have also been proposed. The general idea is to partition the s-dimensional unit hypercube $[0,1)^s$ in k boxes of equal size (usually, $k = d^s$ subcubes), generate n points in the unit hypercube using s uniform random numbers for each point, count how many fall in each box, and then select a way to measure how well these counts agree with the assumption that the points are independent and uniformly distributed over the hypercube. One measure is the chi-square test statistic, but there are many more. A large family of divergence measures includes the chi-square, log-likelihood,

and entropy, among others (Read and Cressie 1988). The *collision test* developed by Knuth (1981) based on ideas from H. D. Christiansen in an unpublished manuscript from 1975, counts the number of times a point falls in a box that already had a point. It tends to be more powerful than the chi-square to detect non-uniformity of the points, because one can use a much larger number of boxes: the expected number of collisions is approximately $n^2/(2k)$ for large k, so one can take k of the order of n^2, whereas for the chi-square k must be $\mathcal{O}(n)$. Marsaglia (1985, 1993) and Marsaglia and Zaman (1993) proposed other specific cases (OPSO, monkey, CAT, ...), some of which count the number of empty boxes instead of the number of collisions (this is equivalent). All these serial-type tests have been studied and compared in a general unifying framework by L'Ecuyer et al. (2002), in both the non-overlapping and overlapping cases. Different asymptotic distributions (Poisson, normal, chi-square, ...) are obtained for the sparse case (n increases more slowly than k), the dense case (k increases more slowly or is fixed), and some intermediate cases.

The *birthday spacings* test proposed by Marsaglia (1985) and further studied by Knuth (1998) and L'Ecuyer and Simard (2001) is a clever variant aimed at detecting spatial regularity of the points produced by the RNG. The boxes are numbered from 0 to $k-1$, the n box numbers that contain the points are sorted in increasing order, then the spacings between those successive numbers are computed and sorted, and the test statistic is the number of collisions between those spacings. The expected number of collisions is approximately $n^3/(4k)$ when k is large, so k should be $\mathcal{O}(n^3)$. L'Ecuyer and Simard (2001) have shown that any LCG with modulus m fails this test when $n \approx 16m^{-1/3}$; i.e., with a number of points in the order of the cubic root of the period length.

Another set of tests have been developed to detect linear dependence in a long sequence of bits. They include random walk tests based on binary sequences (Vattulainen et al. 1994, Shchur et al. 1997), tests on the rank of a random binary matrix (Marsaglia 1985), and tests on the linear complexity of a binary sequence (Carter 1989, Erdmann 1992). These tests are very powerful to detect the linear structure in all RNGs based on linear recurrences modulo 2 discussed in Section 6.

Computer software in which selections (or suites) of predefined statistical tests can be applied together to test an RNG has been proposed in recent decades. Notable tools include TESTRAND of Dudewicz and Ralley (1981), DIEHARD of Marsaglia (1996), the NIST battery (Rukhin et al. 2001, Bassham III et al. 2010), and TestU01 of L'Ecuyer and Simard (2007, 2013). The NIST test suite was designed primarily for cryptographic RNGs. Sýs et al. (2016) give an improved implementation. TestU01 is a large C library that implements an extensive collection of statistical tests and RNGs, various predefined test suites for sequences of $U(0,1)$ random numbers and for sequences of random bits, and several related tools. Its development started before 1990 and early versions were used for testing experiments reported by L'Ecuyer (1992). It also provides facilities to study the interaction between a given test and the structure of the points produced by a given family of RNGs. It permits one to assess at what sample size of the test, as a function of the generators period, all generators from the selected family will start to fail the test systematically. For full period LCGs with prime modulus m and good lattice structure, for example, the required sample size for systematic failure is $\mathcal{O}(m^{1/2})$ for the nearest-pair test and $\mathcal{O}(m^{1/3})$ for the birthday spacings test. Systematic studies based on these tools were published in L'Ecuyer and Hellekalek (1998), L'Ecuyer (2001), and L'Ecuyer and Simard (2001).

5 LINEAR RECURRENCES MODULO A LARGE PRIME

Derrick H. Lehmer was a number theorist who got interested in electronic computers largely because he thought they were fabulous tools for empirical work in pure mathematics, and particularly in number theory. He got involved in the ENIAC project from 1945, quickly became a heavy user of this computer for his work in number theory, and ended up applying his knowledge of number theory to design the first reliable algorithmic RNGs for simulation. In Lehmer (1951), he first stated a general recurrence of order k, $x_i = f(x_{i-1}, \ldots, x_{i-k})$, as a basis for the design of an algorithmic generator, then he settled for the

multiplicative linear recurrence of order 1,

$$x_i = ax_{i-1} \bmod m.$$

This would be good and simple enough for the ENIAC computer and would guarantee a period of $m-1$ by taking m prime and a a primitive element modulo m. To simulate uniform random numbers over the interval $(0,1)$, one would simply take $u_i = x_i/m$. Already in 1950, several people started taking m as a power of 2 for binary computers and a power of 10 for decimal computers, because computing the product modulo m could be done much faster. The maximal achievable period is then only $m/4$ when m is a power of 2 (achieved when $a \bmod 8 = 3$ or 5), and $m/200$ for a power of 10, but this was considered good enough. Taussky and Todd (1954) report specific cases with $m = 2^{43}$, 2^{42}, 2^{39}, 2^{36}, 10^{11}, and 10^{10}, which were in use on various computers at the time. All these generators passed a (small) suite of empirical tests (with sample sizes of a few thousands). These authors also pointed out that when m is a power of 2, the period is maximal, and x_i is written in binary form, then its first bit has period $m/4$, its second bit has period $m/8$, its third bit has period $m/16$, etc., and its last two bits are always the same. That is, the least significant bits have a very short period. The authors say that this can be acceptable because in most practical applications, only the most significant bits are really relevant. This statement is certainly questionable. For example, to generate a random integer in $\{0,1,2,3\}$, an unwary user in search of speed would be tempted to take $x_i \bmod 4$ (the last two bits of x_i), which are always the same!

Rotenberg (1960) proved that the recurrence

$$x_i = (ax_{i-1} + c) \bmod m,$$

with $m = 2^{35}$, $a = 2^d + 1$, and c odd, has full period 2^{35}, and proposed it (with $a = 7$ and $c = 1$) for the IBM 704 35-bit computer. Hull and Dobell (1962) gave maximal-period conditions for general m, a, and $c > 0$, and proved their sufficiency. The conditions are: (i) c is relative prime to m, (ii) every prime divisor of m is also a prime divisor of $a-1$, and (iii) if m is a multiple of 4 then $a-1$ is also a multiple of 4. If m is a power of 2, it suffices to have $a \bmod 4 = 1$ and c odd. They recommended this special case for binary computers. The conditions for this special case were proved earlier by Greenberger (1961).

For $c = 0$, Hull and Dobell (1962) give an expression for the maximal possible period for any integer $m > 0$ and they give conditions on a and x_0 for this maximal period to be reached. These conditions follow from standard results in number theory known at the time (Ore 1948); they were already proved by Gauss in 1801. See Knuth (1998), Section 3.2.1.2, Theorem B. The largest possible period is $m-1$ and is achieved if and only if m is prime and a is a primitive element modulo m.

For a full-period LCG, with period $m-1$ or m, the values $u_i = x_i/m$ visited over the period cover the interval $(0,1)$ or $[0,1)$ very evenly. But what about the independence of successive output values? Aside from empirical statistical tests, from about 1950 to 1967, the theoretical study of the multivariate uniformity of LCGs consisted mainly in developing expressions for the linear correlation of lag r, between u_i and u_{i+r}, over the entire period, for $r \geq 1$. Detailed studies can be found in Jansson (1966) and Knuth (1969), for example. However, linear correlation has limited power to detect dependence in general, and more particularly to detect bad structure in LCGs.

A breakthrough article of Coveyou and MacPherson (1967), released in 1965 as Report ORNL-3866 by the Oak Ridge National Laboratory, introduced the *spectral test* for LCGs, a much more powerful tool than linear correlation to measure their multivariate uniformity. To explain the main idea, we take a multiplicative LCG, to simplify the notation. It turns out that any vector $\mathbf{u} = (u_i, \ldots, u_{i+s-1})$ of s successive output values produced from any initial state belongs to the lattice $L = \{\mathbf{v} = \sum_{j=1}^{s} z_j \mathbf{v}_j : z_j \in \mathbb{Z}\}$, generated by the basis vectors $\mathbf{v}_1 = (1, a, \ldots, a^{s-1})/m$ and $\mathbf{v}_j = \mathbf{e}_j$ (the jth unit vector) for $j = 2, \ldots, s$. All the points of such a lattice belong to equidistant parallel hyperplanes in the s-dimensional space. Coveyou and MacPherson (1967) suggested a frequency analysis that finds the *wavelength* (the distance between successive hyperplanes) for the family of hyperplanes that gives the largest wavelength among those that

contain all the points. The inverse wavelength (the *wave number*, or frequency) is actually the (Euclidean) length of the shortest nonzero vector in the dual lattice. We do not want this vector to be very short, because this implies a large distance between successive hyperplanes, i.e., large slices of space that never contain any points. Coveyou and MacPherson (1967) proposed an algorithm to compute a shortest vector, by first computing bounds on each z_j that must hold if $\mathbf{v}$ is a shortest nonzero vector, and then checking the length of $\mathbf{v}$ for all vectors $(z_1,\ldots,z_s)$ that satisfy the bounds. Since the bounds are tighter when the basis vectors are shorter, they first try to reduce the lengths of the basis vectors by applying simple unimodular transformations to the basis (in a heuristic way). They also observed that the length ℓ_s of the shortest vector in the dual lattice in s dimensions cannot exceed $\gamma_s^{1/2} m^{-1/s}$, where the best possible constant γ_s for a general lattice is known and proved only for $s \le 8$ (still true today) but tight upper bounds are available for $s > 8$ (Conway and Sloane 1999, L'Ecuyer 1999b). They suggested using ℓ_s as a figure of merit and comparing it with the upper bound. They did that for up to $s = 10$ for several LCGs with modulus m near 10^{10} (for decimal computers), and near 2^{31}, 2^{35}, and 2^{47} (for 31-bit, 35-bit, and 47-bit binary computers). They showed that certain LCGs designed to be fast and to optimize the linear correlation between pairs of successive values performed extremely poorly in the spectral test. In particular, a class of LCGs with modulus $m = 2^e$ and a equal to a sum of three powers of 2 (like $a = 2^d + 3$ or $a = 2^d + 5$) were popular at the time, because on a e-bit (or more) computer the product by a modulo m could be done very fast (multiplication by each power of two amounts to a shift and mod m can be done just by masking some bits). This class included (among others) RANDU, with $e = 31$ and $a = 2^{16} + 3$, used on the IBM 360 systems at the time, and RANNO, with $e = 35$ and $a = 2^{18} + 3$, used on the IBM 704 at Yale University. There LCGs all behaved very badly in the spectral test in three or more dimensions, and this could be explained by the fact that all triples of successive values satisfy the recurrence $u_i = 6u_{i-1} - 9u_{i-2} \bmod 1$, which implies that the short vector $(9,-6,1)$ belongs to the dual lattice in three dimensions. Greenberger (1965) pointed out this particular recurrence earlier for RANNO, after observing (with Joseph Lach) "slices of empty space" when looking on an oscilloscope at the points that fall in a thin slice of the unit cube. At about the same time, MacLaren and Marsaglia (1965) observed that these types of LCGs clearly failed empirical statistical tests in three or more dimensions.

Knuth (1969) explained the spectral test of Coveyou and MacPherson (1967) and improved the algorithm. Marsaglia (1968) also discussed the lattice structure of LCGs by focusing on the number of hyperplanes that contain all the points $\mathbf{u}$ instead of the distance between them. This number can be found by computing the length of the shortest nonzero dual lattice vector using the L^1 norm instead of the Euclidean one. In particular, if an LCG satisfies $u_i = 6u_{i-1} - 9u_{i-2} \bmod 1$, then all those triples must belong to no more than 16 planes in the unit cube. Based on a general result of Minkowski, Marsaglia (1968) showed that for any LCG, the s-dimensional points are always covered by at most $(s!m)^{1/s}$ parallel hyperplanes. For $m = 2^{32}$ and $s = 3$, this upper bound is about $(6 \times 2^{32})^{1/3} \approx 2953$, which is much more than 16. By taking a much larger m, e.g., around 2^{200} or more, or by taking a linear recurrence of higher order as we shall see later, this upper bound can be made very large and the distance between hyperplanes can also be made very small. When those papers on the lattice structure were published, some thought it meant the death of LCGs. But from another viewpoint, if we have an LCG with very long period and good lattice structure, whose shortest dual vector is close to its upper bound in a range of dimensions, this gives a certain guarantee of uniformity, which can be arguably safer than a more complicated nonlinear generator for which the multivariate uniformity is too complicated to analyze mathematically, so one can only rely on empirical tests (Niederreiter 1978, Knuth 1969). This viewpoint encouraged the construction of LCGs based on their performance in the spectral test.

Dieter (1975) generalized the algorithm to calculate a shortest nonzero vector in a lattice, with L^1, L^2, and sup norms, and with further improvements compared with Knuth (1969). His algorithm uses both the lattice basis and its dual to reduce the vector lengths by trying to reduce the vectors in pairs until this is no longer possible, and then finds the shortest vector via a branch-and-bound procedure. It can find the distance between hyperplanes as well the number of hyperplanes. This is the algorithm explained in the second and

third editions of Knuth (1981). Fincke and Pohst (1985) came up with a significant improvement to this algorithm by proposing a way to obtain much tighter bounds for each z_j in the branch-and-bound, via a Cholesky decomposition of the matrix of scalar products at each level. We implemented this algorithm in Modula-2 in my lab around 1988.

Lewis et al. (1969) proposed an LCG with prime $m = 2^{31} - 1$ and $a = 16807$ to replace the infamous RANDU on the IBM System/360. This LCG has been very popular in simulation books (Law and Kelton 1982, Bratley et al. 1983) and was the default RNG in prominent simulation software until recently. Several articles and web sites call it the "Park-Miller generator," which is incorrect and misleading. Park and Miller (1988) only mentioned it as a possible minimal standard, arguing that poorer generators (at the time) should be eliminated; it is not their generator. Marsaglia (1972) proposed $a = 69069$ as a "candidate for the best possible multiplier" for $m = 2^{32}$, based on its good lattice structure in up to 5 dimensions and because "it is easy to remember". But this LCG turns out to have a bad lattice structure in 6 dimensions (Knuth 1998).

In the 1970's and 1980's, LCGs were still the most popular RNGs in practice, and it became customary to build or test them with the spectral test. Fishman and Moore III (1986) proposed to divide the length ℓ_s of the shortest vector by its best upper bound, to obtain a value $S_s \in (0,1]$ for each s, and to take $M_t = \min_{s \le t} S_s$ as a figure of merit. They made an extensive search for all the multipliers a for which $M_6 \ge 0.8$, for $m = 2^{31} - 1$. Fishman (1990) made a similar search for $m = 2^{32}$ and 2^{48}. L'Ecuyer adopted this figure of merit M_t in several articles since 1986, for t up to 32, and even more in a few cases.

Lists of several widely-used LCGs with moduli ranging from 2^{24} to 2^{64}, with appropriate references and test results, can be found for example in Table I of L'Ecuyer (1988), in the documentation of module `ulcg` in L'Ecuyer and Simard (2013), with empirical test results in Table I of L'Ecuyer and Simard (2007), and with spectral test results in Table I of Section 3.3.4 of Knuth (1998) and in L'Ecuyer (1999b). LCGs in this last paper were proposed for quasi-Monte Carlo integration and not as RNGs.

For the early LCGs, the multipliers were selected to be small or a sum of a few powers or 2, to make the implementation easier and faster. But those LCGs tend to perform very poorly in the spectral test. Theoretical reasons for this have been given by Couture and L'Ecuyer (1994) and L'Ecuyer (1997, 1999c), in the form of (small) upper bounds on ℓ_s which guarantee bad behavior in the spectral test. For example, Lehmer's original LCG with $a = 23$ performs poorly in the spectral test (Knuth 1998).

To increase the period and (more importantly) to obtain a denser lattice which covers the space more evenly, one can increase the modulus m. But an LCG with a very large m, say larger than 2^{64}, is typically too slow because then the product modulo m cannot be computed directly with standard integer arithmetic on the computer. An easier way to increase the period is to use the higher-order linear recurrence

$$x_i = (a_1 x_{i-1} + \cdots + a_k x_{i-k}) \bmod m, \tag{1}$$

for some integer $k \ge 1$ (the order). This type of recurrence was analyzed by Zierler (1959), who mentioned is potential used as an RNG and gave conditions for having a maximal period of $m^k - 1$ (m must be prime and the characteristic polynomial of the recurrence must be primitive modulo m), among many other results. Alanen and Knuth (1964) proposed a practical way to verify these conditions. An RNG based on this recurrence and output $u_i = x_i/m$ is known as a *multiple recursive generator* (MRG) (Grube 1973, Niederreiter 1978, Lidl and Niederreiter 1986, L'Ecuyer 1990, L'Ecuyer 1994).

A full-period MRG of order k is always k-distributed in base m (if we add the point $(0,\ldots,0)$ to the set of k-dimensional points produced over the main cycle). But in $s > k$ dimensions they have a non-trivial lattice structure whose quality depends on the parameters, just like for LCGs. Coveyou and MacPherson (1967) already introduced the spectral test in the general setting of an MRG of order k. An implementation of the spectral test for general MRGs, for prime or composite m, was given by L'Ecuyer and Couture (1997), under the name of LatMRG, written in Modula-2. This software is currently under reconstruction in C++ using the NTL library. When the period is not maximal, the lattice generated by a single cycle of the MRG can be a strict sublattice of the full lattice generated by all cycles. Couture and L'Ecuyer (1996) showed how to construct a basis for this sublattice and this was also implemented in LatMRG. If one selects

s lacunary indices $0 = i_1 < i_2 < \cdots < i_s$, the points of the form $(u_{i+i_1}, \ldots, u_{i+i_s})$ produced by the MRG over all cycles or over one cycle also generate a lattice that can be analyzed in the same way as L_s, and LatMRG can do that for arbitrary indices. Looking at lacunary indices is often very insightful, because the corresponding lattices are sometimes very bad and this can explain very poor statistical behavior (L'Ecuyer 1997).

Duparc et al. (1953) already studied a special case of the MRG with $k = 2$, based on the Fibonacci recurrence $x_i = (x_{i-1} + x_{i-2}) \bmod m$. They studied the periodicity as a function of m, for any m. This has been generalized to additive *lagged-Fibonacci* generators, whose recurrence has the form

$$x_i \quad = \quad (\pm x_{i-r} \pm x_{i-k}) \bmod m. \tag{2}$$

A specific one was proposed by Mitchell and Moore (1958) (unpublished) with $x_i = (x_{i-24} + x_{i-55}) \bmod m$, m an even number, and $(x_{-54}, \ldots, x_0)$ not all even. The maximal period of an additive lagged-Fibonacci generator is $m^k - 1$ when m is prime and $(2^k - 1)m/2$ when $m = 2^e$ (Brent 1994). Marsaglia (1985) suggested another instance based on $x_i = (x_{i-5} + x_{i-17}) \mod 2^{24}$. The popular Boost Random Number Library (Maurer and Watanabe 2017) still offers nine different Lagged-Fibonacci generators, with periods ranging from 2^{32000} to the astronomical $2^{2300000}$ (so they all have a very large state). However, Lagged-Fibonacci generators in general are known to fail miserably several simple statistical tests (L'Ecuyer 1997), due to their very bad lattice structure. In particular, it is well known that all the points of the form (u_{i-k}, u_{i-r}, u_i) belong to at most three parallel planes defined by $-u_i + u_{i-r} \pm u_{i-k} = c$ for $c \in \{-1, 0, 1\}$ (Couture and L'Ecuyer 1994, L'Ecuyer 1997), and this property shows up in simple empirical tests. This is a simple example where looking at lacunary indices is relevant.

Grube (1973) constructed the first MRGs of order $k > 1$ based on the spectral test. He proposed MRGs of orders 1, 2, and 3, with $m = 2^{31} - 1$. These MRGs have not been used very much until about 20 years later. L'Ecuyer and Blouin (1988) and L'Ecuyer et al. (1993) made computer searches for good full-period MRGs of various orders with prime moduli, based on the spectral test, and proposed implementations. Deng and Lin (2000) proposed a class of fast long-period MRG with a recurrence of the form $x_i = (-x_{i-1} + bx_{i-k}) \bmod m$ for some integer b. Deng and Xu (2003) proposed another class in which the nonzero coefficients a_j are all equal to the same integer b, so that a single multiplication mod m suffices to implement the recurrence. However, L'Ecuyer and Touzin (2004) showed that with these forms of recurrences and a large k, the lattice structure in more than k dimensions cannot be good. This shows up in standard statistical tests (L'Ecuyer and Simard 2007). In a half-dozen other papers published from 2004 to 2012, Deng and his co-authors proposed other variants of these types of long-period MRGs with special structure that make them run fast. They are summarized in L'Ecuyer and Simard (2014), where it is also proved that all of them have a poor lattice structure for certain lacunary indices in more than k dimensions (because the dual lattice always contains a short nonzero vector) and that for certain initial states, they fail empirical statistical tests because their recurrence does not make a sufficiently complicated modification of the state at each step; i.e., their *diffusion capacity* is too limited.

The recurrence (1) can be further generalized to a linear recurrence in matrix form:

$$\mathbf{x}_i = (\mathbf{A}\mathbf{x}_{i-1} + \mathbf{c}) \bmod m,$$

where $\mathbf{A}$ is a $k \times k$ matrix, while $\mathbf{c}$ and the state $\mathbf{x}_i$ are k-dimensional column vectors. At each step, it returns a k-dimensional vector $\mathbf{x}_i/m$. This model was introduced by Niederreiter (1986) and further studied by Afflerbach and Grothe (1988) and Grothe (1987), who adapted the spectral test to these types of generators. See also L'Ecuyer (1990) for further discussion. When m is prime and $\mathbf{c} = \mathbf{0}$ (the most recommendable case in practice) the maximal period is $m^k - 1$ and is reached if and only if the characteristic polynomial of $\mathbf{A}$ is primitive modulo m. Then, each coordinate of $\mathbf{x}_i$ follows exactly the same recurrence, with this characteristic polynomial, but with different starting points. The MRG in (1) is a variant that can be obtained with a particular choice of $\mathbf{A}$ and by returning only the first coordinate of $\mathbf{x}_i/m$ at each step (L'Ecuyer 1990). One can easily jump ahead from $\mathbf{x}_i$ to $\mathbf{x}_{i+\nu}$ for a very large ν with this general matrix recurrence

by first precomputing $\mathbf{A}^\nu \bmod m$ in $\mathcal{O}(\log \nu)$ time via a divide-and-conquer strategy (L'Ecuyer 1990), and then performing a simple matrix-vector multiplication $\mathbf{x}_{i+\nu} = (\mathbf{A}^\nu \bmod m)\mathbf{x}_i \bmod m$ for each vector $\mathbf{x}_i$ of interest. LCGs can also be defined (or represented) in spaces of polynomials, in spaces of formal series, etc. This is sometimes useful for their theoretical analysis and for jumping ahead. See L'Ecuyer (1994).

The MRG recurrence (1) could be computed faster if m was a power of 2, but then the period cannot be very long. Motivated by that, Marsaglia and Zaman (1991) introduced a class of linear recurrences named *add-with-carry* (AWC) and *subtract-with-borrow* (SWB), with the general form

$$\begin{aligned} x_i &= (\pm x_{i-r} + x_{i-k} \pm c_{i-1}) \bmod b, \\ c_i &= \mathbb{I}[\pm x_{i-r} + x_{i-k} \pm c_{i-1} \geq b], \end{aligned}$$

where $\mathbb{I}$ denotes the indicator function, $c_i \in \{0,1\}$ is the carry or borrow at step i, and the period can reach $m-1$ under some conditions, where $m = b^k \pm b^r \mp 1$, even if b is not prime. They recommended specific instances with long periods, e.g., one with $b = 2^{32}$ and $k = 21$ who has 192 cycles of period near $2^{666}/3$, and another one with $b = 2^{32} - 5$ and $k = 43$, with full period $m - 1 = 2^{1376} - 2^{704} + 1$. These generators are very fast and have an extremely long period, so they quickly became very popular. They were recommended by James (1990) and adopted at the CERN research center around that time, and one of them became the default RNG in Mathematica, for example. But it was soon discovered that they gave wrong results in certain types of simulations in physics, related to random walks (Ferrenberg et al. 1992). Soon after that, Tezuka et al. (1993) and Couture and L'Ecuyer (1994) proved that this type of RNG is approximately equivalent (with a difference smaller than $1/b$ in the output) to an LCG with multiplier $a = b^{-1} \bmod m$ (the multiplicative inverse) and modulus $m = b^k \pm b^r \mp 1$, and also that the three-dimensional vectors (u_i, u_{i+r}, u_{i+k}) produced by those LCGs lie in at most three parallel planes, at distance $1/\sqrt{3}$ apart, just like for the lagged-Fibonacci generators. Because of this, they fail several standard empirical statistical tests (L'Ecuyer and Simard 2007). Lüscher (1994) proposed to solve this problem by skipping several values after each block of k, so that the bad vectors are no longer visible. This was implemented by James (1994) in a popular generator named RANLUX, in which a *luxury level* selected by the user determines the amount of skipping. A significant drawback is that the skipping slows down the generator. Boost (Maurer and Watanabe 2017) offers several RANLUX generators and they are quite slow.

This type of linear recurrence with carry was generalized by Koç (1995) (written in 1993) and promoted by Marsaglia (1994), under the name *multiply-with-carry* (MWC). It was further studied by Couture and L'Ecuyer (1997) and Goresky and Klapper (2003). The general MWC recurrence can be written as

$$x_i = (a_1 x_{i-1} + \cdots + a_k x_{i-k} + c_{i-1}) d \mod b, \tag{3}$$

$$c_i = \lfloor (a_0 x_i + a_1 x_{i-1} + \cdots + a_k x_{i-k} + c_{i-1})/b \rfloor, \tag{4}$$

$$u_i = \sum_{\ell=1}^{\infty} x_{i-\ell+1} b^{-\ell}, \tag{5}$$

where b is a positive integer, $a_0, \ldots, a_k$ are arbitrary integers such that a_0 is relatively prime to b, and $d = (-a_0)^{-1} \bmod b$. Couture and L'Ecuyer (1997) showed that this generator is equivalent to an LCG with modulus $m = \sum_{\ell=0}^{k} a_\ell b^\ell$ and multiplier $a = b^{-1} \bmod m$, and that the corresponding ℓ_s in the dual lattice satisfies $\ell_s^2 \leq a_0^2 + \cdots + a_k^2$ for $s \geq k$, so the lattice structure cannot be good unless this sum of squares of coefficients a_j is large. Couture and L'Ecuyer (1997) assumed $-a_0 = d = 1$, which implies that the period cannot exceed $(m-1)/2$ if b is a power of two. Goresky and Klapper (2003) showed that a maximal period of $\rho = m - 1$ can be achieved with a more general a_0, and gave specific instances with maximal period for b ranging from 2^{21} to 2^{35} and ρ up to near 2^{2521}. The MWC can be seen as an efficient way of obtaining a fast RNG with very long period and good quality.

A different way to increase the period and make the structure more complicated at the same time, is to combine several recurrences. MacLaren and Marsaglia (1965) proposed a shuffling technique in which one LCG (with $m = 2^{35}$) is used to select one of the 128 entries in a table as the next output, using its

7 most significant bits, and another one generates a new value to replace this selected entry in the table. They observed a better statistical behavior (empirically) than with one LCG alone. Other variants have been proposed; see, e.g., Marsaglia and Bray (1968), Nance and Overstreet (1978). For example, two recurrences $\{x_i, i \geq 0\}$ and $\{y_i, i \geq 0\}$ can be combined using some binary operation $\circ$ to obtain $z_i = x_i \circ y_i$ at step i. In the case of genuine random variables, it has been proved under mild conditions that this improves the uniformity (Horton and Smith III 1949, Brown and Solomon 1979) and this has often been used as a "proof" that combination improves the uniformity of RNGs (Marsaglia 1985, Deng and Chu 1991, Deng et al. 1997), but this argument is does not stand, because algorithmic RNGs produce deterministic sequences and not independent random variables. Theoretical proofs that combination is better must be based on a clear understanding of the actual recurrences.

Wichmann and Hill (1982) proposed a combination of three small LCGs for 16-bit microcomputers, by adding their outputs modulo 1, and showed that its period was the least common multiple of the individual periods, which is near 2^{43}. Using the Chinese Remainder Theorem, Zeisel (1986) observed that the resulting combination was equivalent to an LCG with modulus equal to the product of the individual moduli (which is not a prime number). This old generator is still the default in Microsoft Excel, despite the fact that it badly fails standard batteries of empirical tests (L'Ecuyer and Simard 2007).

L'Ecuyer (1986, 1988) introduced a different combination method defined as follows. Take J full-period LCGs with prime moduli $m_1, \ldots, m_J$, let $x_{j,i}$ be the state of the jth at step i, and define the output by

$$z_i = (\delta_1 x_{1,i} + \cdots + \delta_J x_{J,i}) \mod m_1, \qquad u_i = z_i / m_1, \tag{6}$$

where $\delta_1, \ldots, \delta_J$ are arbitrary integers. He recommended specific instances with $J = 2$ and $\delta_1 = -\delta_2 = 1$ for 32-bit computers in the papers just cited, and for 16-bit computers in L'Ecuyer (1987), based on a spectral-test analysis of the individual components (not of the resulting generator). In fact, this combined RNG does not have exactly a lattice structure. However, L'Ecuyer and Tezuka (1991) showed that it is approximately equivalent to the following generalization of the Wichmann-Hill combination:

$$w_i = (\delta_1 x_{1,i}/m_1 + \cdots + \delta_J x_{J,i}/m_J) \mod 1, \tag{7}$$

by giving a tight bound on $\sup_i |w_i - u_i|$. They also showed that the combination (7) is exactly equivalent to an LCG with modulus $m = m_1 \cdots m_J$. One can then analyze the multivariate uniformity of the points produced by (6) by applying the spectral test to this LCG. They gave examples of that. L'Ecuyer and Andres (1997) proposed a combined LCG with $J = 4$ components, constructed based on the spectral test and the availability of an efficient implementation.

L'Ecuyer (1996a) generalized the two combination methods (6) and (7) to MRGs of order $k \geq 1$. He showed that combining J MRGs of order k and moduli m_j via (7) is equivalent to another MRG of order k with modulus $m = m_1 \cdots m_J$, and approximately equivalent if it is via (6). L'Ecuyer (1999a) then offered tables of specific parameters and implementations for various J, k, and sizes of the m_j, selected via the spectral test in a search that took several months of CPU time. One of them, named `MRG32k3a`, has $J = 2$, $k = 3$, and the m_j near 2^{32}. It was designed and implemented for computations in 64-bit floating-point arithmetic, at a time when this was more advantageous than using 32-bit integer arithmetic on the (then) popular SUN SparcStations. `MRG32k3a` is now a popular generator, available in plenty of software libraries and systems. Even though it is not one of the fastest available generators, it is reliable. The rationale for these combined MRGs is that one can construct them in a way that each component is easy to implement efficiently (e.g., has just a few nonzero coefficients a_j and they are small) while the combination has a good lattice structure. L'Ecuyer and Touzin (2000) proposed instances for which the nonzero coefficients are sums or differences of just a few powers of 2, which allows faster implementations than `MRG32k3a` with 32-bit integer arithmetic. One of them, named `MRG31k3p`, has $J = 2$, $k = 3$, and the m_j near 2^{31}.

6 LINEAR RECURRENCES MODULO 2

Generators based on linear recurrences modulo $m = 2$ have enjoyed popularity as well, largely because of the binary nature of modern computers. These $\mathbb{F}_2$-linear RNGs (linear in the finite field $\mathbb{F}_2$) can be defined in the following general framework taken from L'Ecuyer and Panneton (2002, 2009):

$$\mathbf{x}_i = \mathbf{A}\mathbf{x}_{i-1}, \tag{8}$$

$$\mathbf{y}_i = \mathbf{B}\mathbf{x}_i, \tag{9}$$

$$u_i = \sum_{\ell=1}^{w} y_{i,\ell-1} 2^{-\ell} = .y_{i,0}\, y_{i,1}\, y_{i,2} \cdots, \tag{10}$$

where the state $\mathbf{x}_i = (x_{i,0}, \ldots, x_{i,k-1})^\mathsf{T} \in \mathbb{F}_2^k$ at step i is a binary column vector of size k, $\mathbf{y}_i = (y_{i,0}, \ldots, y_{i,w-1})^\mathsf{T} \in \mathbb{F}_2^w$ is a binary output vector at step i, $\mathbf{A}$ and $\mathbf{B}$ are binary matrices, all operations in (8) and (9) are performed in $\mathbb{F}_2$ (i.e., modulo 2), and $u_i \in [0,1)$ is the output at step i.

Tausworthe (1965) defined the *Tausworthe* or *linear feedback shift register* (LFSR) generator via

$$x_i = (a_1 x_{i-1} + \cdots + a_k x_{i-k}) \bmod 2, \tag{11}$$

$$u_i = \sum_{\ell=1}^{w} x_{is+\ell-1} 2^{-\ell}$$

where $a_1, \ldots, a_k \in \mathbb{F}_2$, $a_k = 1$, and w and s are positive integers. It takes a block of w successive bits every s steps of the linear recurrence (11) and constructs the output u_i from that. This fits the general $\mathbb{F}_2$-linear framework (8)—(10) by taking $\mathbf{A} = (\mathbf{A}_0)^s$ where

$$\mathbf{A}_0 = \begin{pmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ a_k & a_{k-1} & \ldots & a_1 \end{pmatrix}, \tag{12}$$

in which the blank entries are zeros. Tausworthe (1965) noted that this RNG has maximal period $2^k - 1$ if and only if (11) has a primitive characteristic polynomial modulo 2 and s has no common factor with $2^k - 1$. The name LFSR comes from the fact that (11) can be implemented in hardware via a linear feedback shift register, which is simple and very fast. Tausworthe (1965) proved several desirable properties for his proposed generators, and in particular that the output sequence is *t-distributed to ℓ bits of accuracy* for all $\ell \le s$ and $t \le \lfloor k/s \rfloor$, i.e., that when taking the first ℓ bits of t successive output values, over the entire period, the vector of these $t\ell$ bits take all $2^{t\ell}$ possible values exactly $2^{k-t\ell}$ times, except for the all-zero vector which occurs one time less. To test this *equidistribution* property for any ℓ and t, for any $\mathbb{F}_2$-linear generator, it suffices to write the $t\ell \times t\ell$ binary matrix $\mathbf{M}$ that expresses the $t\ell$ bits involved in the property as linear combinations of the k bits of the initial state, and check if this matrix has full rank (L'Ecuyer 1996b, L'Ecuyer and Panneton 2009).

The early LFSR generators implemented in software were based on primitive trinomials, with only two nonzero coefficients in (11), say a_k and a_r (Tootill et al. 1971, Bright and Enison 1979). An efficient implementation for the case where $k/2 \le r < k \le w$ and $0 < s \le r$ was given by Tezuka and L'Ecuyer (1991) and generalized in L'Ecuyer and Panneton (2009) to more nonzero coefficients.

Tootill et al. (1973) introduced stronger uniformity criteria than what was proved by Tausworthe (1965): the sequence is *asymptotically random* (to w bits) if it is *t-distributed to ℓ bits of accuracy* whenever $\ell \le w$ and $t \le \lfloor k/\ell \rfloor$. They also found an asymptotically random Tausworthe generator with period $2^{607} - 1$, based on the recurrence $x_i = (x_{i-607} + x_{i-273}) \bmod 2$, $s = 512$, and $w = 23$. Asymptotically random is also called *maximally equidistributed* (ME) in many articles and books; e.g., Tezuka (1995), L'Ecuyer (1996b), L'Ecuyer (2012), and is seen as a highly-desirable property.

Lewis and Payne (1973) proposed a fast way to run a generalized version of a Tausworthe generator with two nonzero coefficients $a_k = a_r = 1$, by using a large circular array of bits of size $w \times k$ for the state, and called it a *generalized feedback shift register* (GFSR). At each step, a new column $\mathbf{y}_i$ of the array is computed via $\mathbf{y}_i = (\mathbf{y}_{i-r} \oplus \mathbf{y}_{i-k})$, where each $\mathbf{y}_i$ is a w-bit vector, $\oplus$ is the bitwise exclusive-or (or addition modulo 2), and the array initially contains $\mathbf{y}_{i-k}, \ldots, \mathbf{y}_{i-1})$. The output at step i is constructed as in (10). With this construction, each row of the array runs the same linear sequence (11), but with different starting points which depend on the initial values on that row. Tootill et al. (1973) also used this construction to run a Tausworthe generator by taking initial values so that successive rows are shifted by one step only. Fushimi and Tezuka (1983) showed that the choice of initial state can have a large impact on the equidistribution properties of the GFSR, and proposed a method to construct good initial states.

One drawback of this construction is that it requires a large amount of memory in comparison with the period length: With wk bits for the state we could expect a period near 2^{wk}, but we only get $2^k - 1$. To make the setting more general, suppose the state is an array of $w \times q$ bits (we replace k by q). Since the state has wq bits, the corresponding matrix $\mathbf{A}$ in (8) is $wq \times wq$, and for the GFSR this matrix does not have a primitive characteristic polynomial. Motivated by this, Matsumoto and Kurita (1992) proposed a method that modifies slightly the GFSR recurrence in a way that the corresponding matrix $\mathbf{A}$ has a primitive characteristic polynomial, to obtain a period of $2^k - 1$ with $k = wq$. They called the resulting generator a *twisted GFSR* (TGFSR). Matsumoto and Kurita (1994) pointed out statistical weaknesses in their original TGFSR and proposed an improved version called *tempered TGFSR*, that incorporates *tempering* operations in the matrix $\mathbf{B}$. Matsumoto and Nishimura (1998) later proposed the *Mersenne twister* (MT), a variant of the TGFSR with period equal to a Mersenne prime $2^k - 1$, with k slightly less than wq. Their most famous and widely-used instance is `MT19937`, with $w = 32$, $q = 623$, and period $2^{19937} - 1$. Nishimura (2000) proposed 64-bit instances. Panneton et al. (2006) pointed out weaknesses of the MT. In particular, if `MT19937` starts in (or reaches) a state that has very few ones, it takes up to several hundred thousand steps (on average) before the average output value is approximately 1/2. The explanation is that only a tiny proportion of the 19937 bits of the state are modified at each step. Panneton et al. (2006) developed an $\mathbb{F}_2$-linear class named the WELL generators, with the same period and about the same speed as MT, but with much better equidistribution properties and a matrix $\mathbf{A}$ that changes many more bits of the state at each step. They gave specific parameters and implementations for instances with periods ranging from $2^{512} - 1$ to $2^{44497} - 1$. The proposed WELL are all ME or nearly, whereas the TGFSR and MT cannot be ME in general and the proposed ones are far from ME.

Marsaglia (2003) proposed very fast $\mathbb{F}_2$-linear RNGs whose matrix $\mathbf{A}$ represents a few (e.g., three) successive xorshift operations, where a *xorshift* consists of shifting a block of w bits (usually 32 or 64) in the state by a positions, and xoring it with the original block. Panneton and L'Ecuyer (2005) considered a more general class of xorshift generators, studied maximal-period conditions, limits on the equidistribution, and submitted the generators to statistical testing. They concluded that three-xorshift generators are far from ME and fail many tests, and they proposed versions based on 7 and 13 xorshifts. Brent (2004) proposed generators that combine a xorshift RNG with a Weyl generator. Vigna (2016) proposed to multiply the state of the xorshift generator by a large integer a, modulo $m = 2^{64}$, to produce the output. In both cases, the resulting generators are no longer $\mathbb{F}_2$-linear and they behaves well empirically in statistical tests (L'Ecuyer and Simard 2007).

For a long time, all popular LFSR et GFSR generators were based on primitive trinomials and pentanomials. However Lindholm (1968) and Matsumoto and Kurita (1996) have shown that this leads to poor statistical behavior, and argued that there should be near 50% nonzero coefficients in the characteristic polynomial. The MT generators do not satisfy this, but the WELL do. Tezuka and L'Ecuyer (1991) and Wang and Compagner (1993) proposed a different approach in which two or more LFSR generators are combined via a bitwise exclusive-or of their states (or equivalently of their outputs) and showed that the result is another LFSR whose period is the least common multiple of the periods of the components, and whose characteristic polynomial is the product of those of the components and can have many nonzero

coefficients even if the components are only trinomials (Tezuka 1995). Like for the MRGs, one can then select simple components that can run fast and for which the combination has excellent equidistribution properties and good statistical behavior. L'Ecuyer (1996b, 1999c) provides tables and implementations of ME combined LFSR generators of different sizes. See also Tezuka (1995). This method of combination actually applies to $\mathbb{F}_2$-linear generators in general, as explained in L'Ecuyer and Panneton (2009). L'Ecuyer and Panneton (2002) made concrete constructions of ME combined TGFSR generators with periods around 2^{466} and 2^{1250}.

7 NONLINEAR GENERATORS

Some have argued that generators based only on linear transformations produce points whose structure is too regular and that this can be detected by statistical tests. For example, for any $\mathbb{F}_2$-linear generator, any bit in the output follows a linear recurrence in $\mathbb{F}_2$ that can be recovered from the output, so all these the generators fail a test that measures the linear complexity of a long-enough portion of this binary sequence (L'Ecuyer and Simard 2007). This linearity might be a problem in some situations.

Nonlinear generators can be defined in various ways, for example by using a nonlinear transition function f, or a nonlinear output function g, or by combining two or more linear RNGs of different types (e.g., $\mathbb{F}_2$-linear with an MRG), or by shuffling the output of an RNG with another one as mentioned earlier, or by alternating randomly between several streams, etc.

Knuth (1981) analyzed a generator based on a quadratic recurrence modulo m, of the form $x_i = (ax_{i-1}^2 + bx_{i-1} + c) \bmod m$, and gave conditions for a maximal period of m. If $m = 2^e$, then the period is maximal if and only if a is even, $(b-a-1) \bmod 4 = 0$, and c is odd. More general vectorized versions and higher-order quadratic recurrences have also been considered. Earlier, Coveyou (1969) analyzed the case of a general polynomial recurrence $x_i = p(x_{i-1}) \bmod m$ where $p(x)$ is a polynomial of degree k. He proved for example that if $m = 2^e$, $p(0)$ is odd, and for all $x \in \{0,1,2,3\}$ we have $p'(x) = 1$, $p''(x) = 0$, and $p(x+1) = p(x)+1$, then the period is m.

Marsaglia (1985) proposed a multiplicative lagged-Fibonacci generator, with the recurrence $x_i = (x_{i-r} \times x_{i-k}) \bmod m$ (addition is replaced by a product). For $m = 2^e$, the period can reach $(2^k-1)m/8$ by taking the initial values $x_{-k+1}, \ldots, x_0$ all odd and not all congruent to 1 mod 4. These RNGs are fast and perform very well in empirical tests (L'Ecuyer and Simard 2007).

A variety of nonlinear generators that use inversion modulo m (in the finite field $\mathbb{F}_m$ if m is prime) either in the recurrence or in the output function, have been introduced and analyzed in the early 1990's, mostly by Eichenauer, Niederreiter, and some co-authors. See Eichenauer-Herrmann (1993), Eichenauer-Herrmann et al. (1998), Niederreiter and Shparlinski (2002), and the references therein. Maximal-period instances are easy to find and these generators typically perform well in empirical tests. But they have remained largely unused because they are too slow. It is also unclear how to select good parameters based on easily computable measures of uniformity.

Several other nonlinear RNGs come from the world of cryptology. For example, Blum et al. (1986) proposed the BBS generator, based on the quadratic recurrence $x_i = x_{i-1}^2 \bmod m$ and $u_i = x_i \bmod 2^\nu$ to generate ν-bit integers, where m is the product of two very large unknown (random) primes and x_0 a random integer. Under the assumption that there is no polynomial-time algorithm that can factor arbitrary k-bit integers in $\mathcal{O}(k)$ time, they proved that no polynomial-time (in k) statistical test can distinguish the output sequence of their generator from a truly random sequence. However, it is unclear how large k must be in practice to be safe. The BBS generator is also very slow. L'Ecuyer and Proulx (1989) examine the practical aspects and provide further details. RNGs for cryptology are discussed further in Lagarias (1993) and Luby (1996).

A large class of generators use a faster cryptographic encoding method either for the transition f (the *output feedback mode*) or for the output function g. The simplest variant of the second case is when the state is just a counter which is increased by 1 each time the generator is called, and g is an encryption algorithm. There are other variants. Hellekalek and Wegenkittl (2003) studied and compared these methods

with the *advanced encryption standard* (AES) (NIST 2001) for g, and found that they perform quite well. Several other encryption methods have been proposed recently, such as SHA, TEA, ChaCha, Threefish, etc.; see Salmon et al. (2011), L'Ecuyer et al. (2017), and the references given there. Many of them are simpler and less secure than AES and BBS, but they are faster. They have been recently promoted for use on GPUs, which require thousands of small-state generators running in parallel.

Another way of creating nonlinearity at low cost (in speed) is to combine a small nonlinear generator with a fast long-period linear one (Aiello et al. 1998), e.g., by addition modulo 1 or bitwise xor. L'Ecuyer and Granger-Piché (2003) combine an MRG with an $\mathbb{F}_2$-linear generator and they prove that the combined generator has at least as much uniformity (in covering the space) as any of its components These types of combined generators perform very well in empirical tests.

8 MULTIPLE STREAMS AND SUBSTREAMS

A single stream of random numbers is often not sufficient for simulation applications. Multiple streams and substreams that can evolve independently of each other once initialized are very convenient for example to facilitate the synchronization when comparing different systems or different parameters of a system with common random numbers (Law and Kelton 1982, Bratley et al. 1983, L'Ecuyer 2007) and are essential for running simulations on parallel processors (L'Ecuyer 2015, L'Ecuyer et al. 2017).

The most popular and simplest way to obtain a large number of streams is to take a long-period generator and compute seeds that are regularly spaced far from each other in the sequence. For this, we need efficient facilities to jump ahead by ν positions in the sequence even for very large ν, without generating intermediate values. We saw earlier how to do that for linear generators base on a linear recurrence of the form $\mathbf{x}_i = \mathbf{A}\mathbf{x}_{i-1} \bmod m$ (L'Ecuyer 1990).

Bratley et al. (1983) provided a table of seeds spaced 100,000 steps apart for the LCG with $m = 2^{31} - 1$ and $a = 16807$, computed with this approach. L'Ecuyer and Côté (1991) proposed a software tool based on the combined RNG of L'Ecuyer (1988) (whose period is near 2^{61}), with multiple streams spaced 2^{50} steps apart. Each stream was further partitioned into $V = 2^{20}$ substreams of length $W = 2^{30}$. For examples showing the usefulness of substreams, see L'Ecuyer (2015) and L'Ecuyer et al. (2015). The streams were numbered from 1 to G, where G could be selected by the user, with a default value of 32. The code was in Pascal and the states were stored in arrays. This same system was used already in the SIMOD Modula-2 simulation software library (L'Ecuyer and Giroux 1987). A similar system was proposed by L'Ecuyer and Andres (1997), based on a RNG with longer period which permitted many more streams and substreams, and implemented in Modula-2 and C.

L'Ecuyer et al. (2002) proposed an improved tool named `RngStreams`, based on the `MRG32k3a` generator with period near 2^{191}, and in which the streams are objects that can be created in a practically unlimited number. It is implemented in C, C++ and Java. This package is now featured in a large variety of software environments, including SSJ (L'Ecuyer and Buist 2005, L'Ecuyer 2016), R (L'Ecuyer and Leydold 2005, R Core Team 2015), MATLAB, SAS, Arena, etc. SSJ actually offers multiple streams and substreams from several different RNGs, including MRGs, combined LFSRs, MT, WELL, etc. L'Ecuyer et al. (2015) proposed a similar tool offering streams and substreams for parallel computing on GPUs.

For the MT and WELL, the matrix $\mathbf{A}$ is very large, e.g., 19937×19937 bits for `MT19937`, so raising $\mathbf{A}$ to a large power and multiplying it by $\mathbf{x}_i$ in the standard way is much too slow. Haramoto et al. (2008a, 2008b) have proposed faster methods to do that via polynomial arithmetic. In parallel settings where thousands or millions of random streams are required, and more particularly on GPUs, RNGs with such a large state are not appropriate, because of the excessive overhead for initializing these streams and copying them across the different levels of computer memory (L'Ecuyer et al. 2017).

Mascagni and Srinivasan (2000) proposed a different type of software tool with multiple streams defined in a tree structure, for parallel simulation. That is, any stream can create new children streams, at any level, without relying on a central monitor to define the streams. Instead of splitting the period of the RNG in subsequences to define the streams, the authors use different RNG parameters (e.g., LCGs with

different multipliers). Bad parameter choices are detected and screened out by a small battery of empirical tests (no theoretical measures of uniformity are computed). The RNGs considered are lagged Fibonacci, LCGs, and LCGs combined with a fixed MRG. Trees of RNGs have been proposed earlier by Frederickson et al. (1987). Halton (1989) studied trees of LCGs in great details and proposed constructions in which the LCGs have the same m and a but different additive constants c. Durst (1989) observed that all these LCGs produce the same sequence, shifted by different constants. He suggested using a single RNG with independent seeds chosen at random and analyzed the chance of overlap in any two streams.

9 FURTHER INFORMATION

Knuth (1998), Niederreiter (1992) and Tezuka (1995) provide detailed discussions on RNGs. Luby (1996) covers cryptographic generators. Review articles and tutorials include L'Ecuyer (1994, 1998, 2012, 2013, 2015) and L'Ecuyer and Hellekalek (1998). An overview and discussion of RNGs for parallel computing, with several references, can be found in L'Ecuyer et al. (2017). Extended bibliographies on uniform RNGs are given by Sowey (1972) and Beebe (2017). The latter contains 3885 references.

REFERENCES

Afflerbach, L., and H. Grothe. 1988. "The Lattice Structure of Pseudo-Random Vectors Generated by Matrix Generators". *Journal of Computational and Applied Mathematics* 23:127–131.

Aiello, W., S. Rajagopalan, and R. Venkatesan. 1998. "Design of Practical and Provably Good Random Number Generators". *Journal of Algorithms* 29 (2): 358–389.

Alanen, J. D., and D. E. Knuth. 1964. "Tables of Finite Fields". *SANKHYĀ: The Indian Journal of Statistics, Series A* 26:305–328.

André, D. 1884. "Étude sur les maxima, minima et squences des permutations". *Annales Scientifiques de l'École Normale Supérieure* 3 (1): 121–134.

Barton, D. E., and C. L. Mallows. 1965. "Some Aspects of the Random Sequence". *The Annals of Mathematical Statistics* 36 (1): 236–260.

Bassham III, L. E., A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks, N. A. Heckert, J. F. Dray, and S. Vo. 2010. "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications". NIST special pub. 800-22, rev. 1a, National Institute of Standards and Technology, Gaithersburg, MD, USA.

Beebe, N. H. F. 2017. "A Bibliography of Pseudorandom Number Generation, Sampling, Selection, Distribution, and Testing". http://www.math.utah.edu/~beebe/.

Bienaymé, M. 1875. "Application d'un théorème nouveau du calcul des probabilités". *Comptes Rendus des Séances de l'Académie des Sciences de Paris* LXXXI (10): 417–423.

Blum, L., M. Blum, and M. Schub. 1986. "A Simple Unpredictable Pseudo-Random Number Generator". *SIAM Journal on Computing* 15 (2): 364–383.

Blum, M. 1986. "Independent Unbiased Coin Flips From a Correlated Biased Source—A Finite State Markov Chain". *Combinatorica* 6 (2): 97–108.

Borel, E. 1909. "Le continu mathmatique et le continu physique". *Rivista di scienza* 6:21–35.

Bratley, P., B. L. Fox, and L. E. Schrage. 1983. *A Guide to Simulation*. New York, Springer-Verlag.

Brent, R. P. 1994. "On the Periods of Generalized Fibonacci Recurrences". *Mathematics of Computation* 63 (207): 389–401.

Brent, R. P. 2004. "Note on Marsaglia's Xorshift Random Number Generators". *Journal of Statistical Software* 11 (5): 1–4.

Bright, H. S., and R. L. Enison. 1979. "Quasi-Random Number Sequences from a Long-Period TLP Generator with Remarks on Applications to Cryptography". *Computing Surveys* 11:357–370.

Brown, B. B. 1948. "Some Tests on the Randomness of a Million Digits". Technical Report P44, available at http://www.rand.org/pubs/papers/P44.html, RAND Corporation, Santa Monica, CA.

Brown, G. W. 1949. "History of RAND's Random Digits: Summary". Technical Report P113, available at http://www.rand.org/pubs/papers/P113.html, RAND Corporation, Santa Monica, CA.

Brown, M., and H. Solomon. 1979. "On Combining Pseudorandom Number Generators". *Annals of Statistics* 1:691–695.

Carr, K. 2017. "History of Dice". http://quatr.us/games/dice.htm.

Carter, G. D. 1989. *Aspects of Local Linear Complexity*. Ph. D. thesis, University of London.

Chor, B., and O. Goldreich. 1988. "Unbiased Bits From Sources of Weak Randomness and Probabilistic Communication Complexity". *SIAM Journal on Computation* 17 (2): 230–261.

Chudnovsky, D., and G. Chudnovsky. 1989, November. "The computation of classical constants". *Proceedings of the National Academy of Sciences of the USA* 86:8178–8182.

Cobine, J. D., and J. R. Curry. 1947. "Electrical Noise Generators". *Proceedings of the Institute of Radio Engineers* 35:875–879.

Conway, J. H., and N. J. A. Sloane. 1999. *Sphere Packings, Lattices and Groups*. 3rd ed. Grundlehren der Mathematischen Wissenschaften 290. New York: Springer-Verlag.

Couture, R., and P. L'Ecuyer. 1994. "On the Lattice Structure of Certain Linear Congruential Sequences Related to AWC/SWB Generators". *Mathematics of Computation* 62 (206): 798–808.

Couture, R., and P. L'Ecuyer. 1996. "Orbits and Lattices for Linear Random Number Generators with Composite Moduli". *Mathematics of Computation* 65 (213): 189–201.

Couture, R., and P. L'Ecuyer. 1997. "Distribution Properties of Multiply-with-Carry Random Number Generators". *Mathematics of Computation* 66 (218): 591–607.

Coveyou, R. R. 1969. "Random number generation is too important to be left to chance". In *Applied Probability and Monte Carlo Methods and modern aspects of dynamics*, Studies in applied mathematics 3, 70–111. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Coveyou, R. R., and R. D. MacPherson. 1967. "Fourier Analysis of Uniform Random Number Generators". *Journal of the ACM* 14:100–119.

Deng, L.-Y., and Y.-C. Chu. 1991. "Combining random number generators". In *Proceedings of the 1991 Winter Simulation Conference*, 1043–1046: IEEE Computer Society.

Deng, L.-Y., and D. K. J. Lin. 2000. "Random Number Generation for the New Century". *The American Statistician* 54 (2): 145–150.

Deng, L.-Y., D. K. J. Lin, J. Wang, and Y. Yuan. 1997. "Statistical Justification of Combination Generators". *Statistica Sinica* 7:993–1003.

Deng, L.-Y., and H. Xu. 2003. "A System of high-dimensional, efficient, long-cycle and portable uniform random number generators". *ACM Transactions on Modeling and Computer Simulation* 13 (4): 299–309.

Devroye, L. 1986. *Non-Uniform Random Variate Generation*. New York, NY: Springer-Verlag.

Dieter, U. 1975. "How to Calculate Shortest Vectors in a Lattice". *Mathematics of Computation* 29 (131): 827–833.

Dudewicz, E. J., and T. G. Ralley. 1981. *The Handbook of Random Number Generation and Testing with TESTRAND Computer Code*. Columbus, Ohio: American Sciences Press.

Duparc, H. J. A., C. G. Lekkerkerker, and W. Peremans. 1953. "Reduced Sequences of Integers and Pseudorandom Numbers". Technical Report ZW 1953-002, Mathematisch Centrum, Amsterdam.

Durst, M. J. 1989. "Using Linear Congruential Generators for Parallel Random Number Generation". In *Proceedings of the 1989 Winter Simulation Conference*, 462–466: IEEE Press.

Eichenauer-Herrmann, J. 1993. "Statistical Independence of a New Class of Inversive Congruential Pseudorandom Numbers". *Mathematics of Computation* 60:375–384.

Eichenauer-Herrmann, J., E. Herrmann, and S. Wegenkittl. 1998. "A Survey of Quadratic and Inversive Congruential Pseudorandom Numbers". In *Monte Carlo and Quasi-Monte Carlo Methods 1996*, edited by P. Hellekalek, G. Larcher, H. Niederreiter, and P. Zinterhof, Volume 127 of *Lecture Notes in Statistics*, 66–97. New York, NY: Springer.

Erdmann, E. D. 1992. "Empirical Tests of Binary Keystreams". Master's thesis, Department of Mathematics, Royal Holloway and Bedford New College, University of London.

Esmenjaud-Bonnardel, M. 1965. "Étude Statistique des Décimales de π". *Revue Française de Techniques Informatiques* 8 (4): 295–306.

Ferrenberg, A. M., D. P. Landau, and Y. J. Wong. 1992. "Monte Carlo Simulations: Hidden Errors From "Good" Random Number Generators". *Physical Review Letters* 69 (23): 3382–3384.

Fincke, U., and M. Pohst. 1985. "Improved Methods for Calculating Vectors of Short Length in a Lattice, Including a Complexity Analysis". *Mathematics of Computation* 44:463–471.

Fisher, R. A. 1926. "Correspondence and Notes: On the Random Sequence". *Quarterly Journal of the Royal Meteorological Society* 52 (219): 250–250.

Fisher, R. A., and F. Yates. 1938. *Statistical tables for biological, agricultural and medical research*. London: Oliver and Boyd.

Fishman, G. S. 1990, Jan. "Multiplicative Congruential Random Number Generators with Modulus 2^{β}: An Exhaustive Analysis for $\beta = 32$ and a Partial Analysis for $\beta = 48$". *Mathematics of Computation* 54 (189): 331–344.

Fishman, G. S., and L. S. Moore III. 1986. "An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus $2^{31} - 1$". *SIAM Journal on Scientific and Statistical Computing* 7 (1): 24–45.

Forsythe, G. E. 1951. "Generation and testing of 1,217,370 "random" binary digits of the SWAC (abstract)". *Bulletin of the American Mathematical Society* 57 (4): 304.

Franklin, J. N. 1963. "Deterministic Simulation of Random Processes". *Mathematics of Computation* 17:28–59.

Franklin, J. N. 1965. "Numerical Simulation of Stationary and Non-Stationary Gaussian Random Processes". *SIAM Review* 7 (1): 68–80.

Frederickson, P., R. Hiromoto, and J. Larson. 1987. "A parallel Monte Carlo transport algorithm using a pseudo-random tree to guarantee reproducibility". *Parallel Computing* 4 (3): 281–290.

Fu, J. C. 1996. "Distribution theory of runs and patterns associated with a sequence of multi-state trials". *Statistica Sinica* 6 (4): 957–974.

Fu, J. C., and M. V. Koutras. 1994. "Distribution Theory of Runs: A Markov Chain Approach". *Journal of the American Statistical Association* 89 (427): 1050–1058.

Fushimi, M., and S. Tezuka. 1983. "The k-Distribution of Generalized Feedback Shift Register Pseudorandom Numbers". *Communications of the ACM* 26 (7): 516–523.

Galton, F. 1890. "Dice for Statistical Experiments". *Nature* 41:13–14.

Glimne, D. 2014 (accessed March 23, 2017). "dice". *Encyclopedia Britannica*.

Good, I. J. 1953. "The Serial Test for Sampling Numbers and Other Tests for Randomness". *Proceedings of the Cambridge Philosophical Society* 49:276–284.

Goresky, M., and A. Klapper. 2003. "Efficient Multiply-with-Carry Random Number Generators with Maximal Period". *ACM Transactions on Modeling and Computer Simulation* 13 (4): 310–321.

Greenberger, M. 1961. "Notes on a New Pseudo-Random Number Generator". *Journal of the ACM* 8 (2): 163–167.

Greenberger, M. 1965. "Method in Randomness". *Communications of the ACM* 8 (3): 177–179.

Greenwood, R. E. 1955. "Coupon collector's test for random digits". *Math. Tables and other Aids to Computation* 9:1–5, 224, 229.

Grothe, H. 1987. "Matrix Generators for Pseudo-Random Vectors Generation". *Statistische Hefte* 28:233–238.

Grube, A. 1973. "Mehrfach rekursiv-erzeugte Pseudo-Zufallszahlen". *Zeitschrift für angewandte Mathematik und Mechanik* 53:T223–T225.

Gruenberger, F., and A. M. Mark. 1951. "The d^2 test of random digits". *Mathematical Tables and Other Aids to Computation* 5 (34): 109–110.

Halton, J. H. 1989. "Pseudo-random Trees: Multiple Independent Sequence Generators for Parallel and Branching Computations". *Journal of Computational Physics* 84:1–56.

Haramoto, H., M. Matsumoto, and P. L'Ecuyer. 2008. "A Fast Jump Ahead Algorithm for Linear Recurrences in a Polynomial Space". In *Proceedings of SETA 2008*, Lectures notes in Computer Science, 385–390.

Haramoto, H., M. Matsumoto, T. Nishimura, F. Panneton, and P. L'Ecuyer. 2008. "Efficient Jump Ahead for $\mathbf{F}_2$-Linear Random Number Generators". *INFORMS Journal on Computing* 20 (3): 290–298.

Hellekalek, P., and S. Wegenkittl. 2003. "Empirical Evidence concerning AES". *ACM Transactions on Modeling and Computer Simulation* 13 (4): 322–333.

Herrero-Collantes, M., and J. C. Garcia-Escartin. 2017. "Quantum random number generators". *Reviews of Modern Physics* 89:015004.

Horton, H. B., and R. T. Smith III. 1949. "A Direct Method for Producing Random Digits in any Number System". *Annals of Mathematical Statistics* 20:82–90.

Hull, T. E., and A. R. Dobell. 1962. "Random Number Generators". *SIAM Review* 4:230–254.

James, F. 1990. "A Review of Pseudorandom Number Generators". *Computer Physics Communications* 60:329–344.

James, F. 1994. "RANLUX: A Fortran Implementation of the High-Quality Pseudorandom Number Generator of Lüscher". *Computer Physics Communications* 79:111–114.

Jansson, B. 1966. *Random Number Generators*. Stockholm: Victor Pettersons Bokindustri Aktiebolag.

Kendall, M. G., and B. Babington-Smith. 1938. "Randomness and other random sampling numbers". *Journal of the Royal Statistical Society* 101:147–166.

Kendall, M. G., and B. Babington-Smith. 1939. "Second paper on random sampling numbers". *Journal of the Royal Statistical Society Supplement* 6:51–61.

Kermack, W. O., and A. G. Kendrick. 1937a. "Some distributions associated with a randomly arranged set of numbers". *Proc. of the Royal Society of Edinburgh* 57:332–376.

Kermack, W. O., and A. G. Kendrick. 1937b. "Tests for randomness in a series of numerical observations". *Proc. of the Royal Society of Edinburgh* 57:228–240.

Knuth, D. E. 1969. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. First ed. Reading, MA: Addison-Wesley.

Knuth, D. E. 1981. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Second ed. Reading, MA: Addison-Wesley.

Knuth, D. E. 1998. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Third ed. Reading, MA: Addison-Wesley.

Koç, C. 1995. "Recurring-with-Carry Sequences". *Journal of Applied Probability* 32:966–971.

Korobov, N. M. 1948. "On Functions with Uniformly Distributed Fractional Parts". *Doklady Akad. Nauk SSSR* 62:21–22.

Kuipers, L., and H. Niederreiter. 1974. *Uniform Distribution of Sequences*. New York, NY: John Wiley.

Lagarias, J. C. 1993. "Pseudorandom Numbers". *Statistical Science* 8 (1): 31–39.

Law, A. M., and W. D. Kelton. 1982. *Simulation Modeling and Analysis*. NY: McGraw-Hill.

L'Ecuyer, P. 1986. "Efficient and Portable 32-Bit Random Variate Generators". In *Proceedings of the 1986 Winter Simulation Conference*, 275–277.

L'Ecuyer, P. 1987. "A Portable Random Number Generator for 16-Bit Computers". In *Modeling and Simulation on Microcomputers 1987*, 45–49. The Society for Computer Simulation.

L'Ecuyer, P. 1988. "Efficient and Portable Combined Random Number Generators". *Communications of the ACM* 31 (6): 742–749 and 774. Correspondence in same journal, 32, 8 (1989) 1019–1024.

L'Ecuyer, P. 1990. "Random Numbers for Simulation". *Communications of the ACM* 33 (10): 85–97.

L'Ecuyer, P. 1992, Dec. "Testing Random Number Generators". In *Proceedings of the 1992 Winter Simulation Conference*, 305–313: IEEE Press.

L'Ecuyer, P. 1994. "Uniform Random Number Generation". *Annals of Operations Research* 53:77–120.

L'Ecuyer, P. 1996a. "Combined Multiple Recursive Random Number Generators". *Operations Research* 44 (5): 816–822.

L'Ecuyer, P. 1996b. "Maximally Equidistributed Combined Tausworthe Generators". *Mathematics of Computation* 65 (213): 203–213.

L'Ecuyer, P. 1997. "Bad Lattice Structures for Vectors of Non-Successive Values Produced by Some Linear Recurrences". *INFORMS Journal on Computing* 9 (1): 57–60.

L'Ecuyer, P. 1998. "Random Number Generation". In *Handbook of Simulation*, edited by J. Banks, 93–137. Wiley. chapter 4.

L'Ecuyer, P. 1999a. "Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators". *Operations Research* 47 (1): 159–164.

L'Ecuyer, P. 1999b. "Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure". *Mathematics of Computation* 68 (225): 249–260. See the Errata at http://www.iro.umontreal.ca/~lecuyer/myftp/papers/latrules99Errata.pdf.

L'Ecuyer, P. 1999c. "Tables of Maximally Equidistributed Combined LFSR Generators". *Mathematics of Computation* 68 (225): 261–269.

L'Ecuyer, P. 2001. "Software for Uniform Random Number Generation: Distinguishing the Good and the Bad". In *Proceedings of the 2001 Winter Simulation Conference*, 95–105. Piscataway, NJ: IEEE Press.

L'Ecuyer, P. 2007. "Variance Reduction's Greatest Hits". In *Proceedings of the 2007 European Simulation and Modeling Conference*, 5–12. Ghent, Belgium: EUROSIS.

L'Ecuyer, P. 2012. "Random Number Generation". In *Handbook of Computational Statistics* (second ed.)., edited by J. E. Gentle, W. Haerdle, and Y. Mori, 35–71. Berlin: Springer-Verlag.

L'Ecuyer, P. 2013. "Random Number Generators". In *Encyclopedia of Operations Research and Management Science* (third ed.)., M. C. Fu and S. I. Gass eds, Chapter 958, 1256–1263. Springer-Verlag.

L'Ecuyer, P. 2015. "Random Number Generation with Multiple Streams for Sequential and Parallel Computers". In *Proceedings of the 2015 Winter Simulation Conference*, 31–44: IEEE Press.

L'Ecuyer, P. 2016. "SSJ: Stochastic Simulation in Java, Software Library". http://simul.iro.umontreal.ca/ssj/.

L'Ecuyer, P., and T. H. Andres. 1997. "A Random Number Generator Based on the Combination of Four LCGs". *Mathematics and Computers in Simulation* 44:99–107.

L'Ecuyer, P., and F. Blouin. 1988. "Linear Congruential Generators of Order $k > 1$". In *Proceedings of the 1988 Winter Simulation Conference*, 432–439: IEEE Press.

L'Ecuyer, P., F. Blouin, and R. Couture. 1993. "A Search for Good Multiple Recursive Random Number Generators". *ACM Transactions on Modeling and Computer Simulation* 3 (2): 87–98.

L'Ecuyer, P., and E. Buist. 2005. "Simulation in Java with SSJ". In *Proceedings of the 2005 Winter Simulation Conference*, edited by M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 611–620. Piscataway, NJ: IEEE Press.

L'Ecuyer, P., J.-F. Cordeau, and R. Simard. 2000. "Close-Point Spatial Tests and their Application to Random Number Generators". *Operations Research* 48 (2): 308–317.

L'Ecuyer, P., and S. Côté. 1991. "Implementing a Random Number Package with Splitting Facilities". *ACM Transactions on Mathematical Software* 17 (1): 98–111.

L'Ecuyer, P., and R. Couture. 1997. "An Implementation of the Lattice and Spectral Tests for Multiple Recursive Linear Random Number Generators". *INFORMS Journal on Computing* 9 (2): 206–217.

L'Ecuyer, P., and N. Giroux. 1987. "A Process-Oriented Simulation Package Based on Modula-2". In *1987 Winter Simulation Proceedings*, 165–174.

L'Ecuyer, P., and J. Granger-Piché. 2003. "Combined Generators with Components from Different Families". *Mathematics and Computers in Simulation* 62:395–404.

L'Ecuyer, P., and P. Hellekalek. 1998. "Random Number Generators: Selection Criteria and Testing". In *Random and Quasi-Random Point Sets*, edited by P. Hellekalek and G. Larcher, Volume 138 of *Lecture Notes in Statistics*, 223–265. New York, NY: Springer-Verlag.

L'Ecuyer, P., and J. Leydold. 2005. *rstream: Streams of Random Numbers for Stochastic Simulation*.

L'Ecuyer, P., D. Munger, and N. Kemerchou. 2015. "clRNG: A Random Number API with Multiple Streams for OpenCL". report, http://www.iro.umontreal.ca/~lecuyer/myftp/papers/clrng-api.pdf.

L'Ecuyer, P., D. Munger, B. Oreshkin, and R. Simard. 2017. "Random Numbers for Parallel Computers: Requirements and Methods, with Emphasis on GPUs". *Mathematics and Computers in Simulation* 135:3–17. Open access at http://dx.doi.org/10.1016/j.matcom.2016.05.005.

L'Ecuyer, P., and F. Panneton. 2002. "Construction of Equidistributed Generators Based on Linear Recurrences Modulo 2". In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, edited by K.-T. Fang, F. J. Hickernell, and H. Niederreiter, 318–330. Berlin: Springer-Verlag.

L'Ecuyer, P., and F. Panneton. 2009. "$\mathbf{F}_2$-Linear Random Number Generators". In *Advancing the Frontiers of Simulation: A Festschrift in Honor of George Samuel Fishman*, edited by C. Alexopoulos, D. Goldsman, and J. R. Wilson, 169–193. New York: Springer-Verlag.

L'Ecuyer, P., and R. Proulx. 1989, Dec. "About Polynomial-Time "Unpredictable" Generators". In *Proceedings of the 1989 Winter Simulation Conference*, 467–476: IEEE Press.

L'Ecuyer, P., and R. Simard. 1999. "Beware of Linear Congruential Generators with Multipliers of the Form $a = \pm 2^q \pm 2^r$". *ACM Transactions on Mathematical Software* 25 (3): 367–374.

L'Ecuyer, P., and R. Simard. 2001. "On the Performance of Birthday Spacings Tests for Certain Families of Random Number Generators". *Mathematics and Computers in Simulation* 55 (1–3): 131–137.

L'Ecuyer, P., and R. Simard. 2007, August. "TestU01: A C Library for Empirical Testing of Random Number Generators". *ACM Transactions on Mathematical Software* 33 (4): Article 22.

L'Ecuyer, P., and R. Simard. 2013. *TestU01: A Software Library in ANSI C for Empirical Testing of Random Number Generators*. Software user's guide, May 16, 2013, http://www.iro.umontreal.ca/~lecuyer.

L'Ecuyer, P., and R. Simard. 2014. "On the Lattice Structure of a Special Class of Multiple Recursive Random Number Generators". *INFORMS Journal on Computing* 26 (2): 449–460.

L'Ecuyer, P., R. Simard, E. J. Chen, and W. D. Kelton. 2002. "An Object-Oriented Random-Number Package with Many Long Streams and Substreams". *Operations Research* 50 (6): 1073–1075.

L'Ecuyer, P., R. Simard, and S. Wegenkittl. 2002. "Sparse Serial Tests of Uniformity for Random Number Generators". *SIAM Journal on Scientific Computing* 24 (2): 652–668.

L'Ecuyer, P., and S. Tezuka. 1991. "Structural Properties for Two Classes of Combined Random Number Generators". *Mathematics of Computation* 57 (196): 735–746.

L'Ecuyer, P., and R. Touzin. 2000. "Fast Combined Multiple Recursive Generators with Multipliers of the Form $a = \pm 2^q \pm 2^r$". In *Proceedings of the 2000 Winter Simulation Conference*, 683–689. Piscataway, NJ: IEEE Press.

L'Ecuyer, P., and R. Touzin. 2004. "On the Deng-Lin Random Number Generators and Related Methods". *Statistics and Computing* 14:5–9.

Lehmer, D. H. 1951. "Mathematical Methods in Large Scale Computing Units". *The Annals of the Computation Laboratory of Harvard University* 26:141–146.

Levene, H., and J. Wolfowitz. 1944. "The Covariance Matrix of Runs Up and Down". *The Annals of Mathematical Statistics* 15 (1): 58–69.

Lewis, P. A. W., A. S. Goodman, and J. M. Miller. 1969. "A Pseudo-random Number Generator for the System/360". *IBM System's Journal* 8:136–143.

Lewis, T. G., and W. H. Payne. 1973. "Generalized Feedback Shift Register Pseudorandom Number Algorithm". *Journal of the ACM* 20 (3): 456–468.

Lidl, R., and H. Niederreiter. 1986. *Introduction to Finite Fields and Their Applications*. Cambridge: Cambridge University Press.

Lindholm, J. H. 1968. "An Analysis of the Pseudo-Randomness Properties of Subsequences of Long m-Sequences". *IEEE Transactions on Information Theory* IT-14 (4): 569–576.

Luby, M. 1996. *Pseudorandomness and Cryptographic Applications*. Princeton: Princeton University Press.

Lüscher, M. 1994. "A Portable High-Quality Random Number Generator for Lattice Field Theory Simulations". *Computer Physics Communications* 79:100–110.

MacLaren, N. M., and G. Marsaglia. 1965. "Uniform Random Number Generators". *Journal of the ACM* 12:83–89.

Marsaglia, G. 1968. "Random Numbers Fall Mainly in the Planes". *Proceedings of the National Academy of Sciences of the United States of America* 60:25–28.

Marsaglia, G. 1972. "The structure of linear congruential sequences". In *Applications of Number Theory to Numerical Analysis*, edited by S. K. Zaremba, 249–285. London: Academic Press.

Marsaglia, G. 1985. "A Current View of Random Number Generators". In *Computer Science and Statistics, Sixteenth Symposium on the Interface*, edited by L. Billard, 3–10. North-Holland, Amsterdam: Elsevier Science Publishers.

Marsaglia, G. 1993. "Remarks on Choosing and Implementing Random Number Generators". *Communications of the ACM* 36 (7): 105–110.

Marsaglia, G. 1994. "Yet Another RNG". Posted to the electronic billboard `sci.stat.math`, August 1.

Marsaglia, G. 1996. "DIEHARD: A Battery of Tests of Randomness". http://www.stat.fsu.edu/pub/diehard.

Marsaglia, G. 2003. "Xorshift RNGs". *Journal of Statistical Software* 8 (14): 1–6.

Marsaglia, G., and T. A. Bray. 1968. "On-Line Random Number Generators and their Use in Combinations". *Communications of the ACM* 11:757–759.

Marsaglia, G., and A. Zaman. 1991. "A New Class of Random Number Generators". *The Annals of Applied Probability* 1:462–480.

Marsaglia, G., and A. Zaman. 1993. "Monkey Tests for Random Number Generators". *Computers Math. Applic.* 26 (9): 1–10.

Mascagni, M., and A. Srinivasan. 2000. "Algorithm 806: SPRNG: A Scalable Library for Pseudorandom Number Generation". *ACM Transactions on Mathematical Software* 26:436–461.

Matsumoto, M., and Y. Kurita. 1992. "Twisted GFSR Generators". *ACM Transactions on Modeling and Computer Simulation* 2 (3): 179–194.

Matsumoto, M., and Y. Kurita. 1994. "Twisted GFSR Generators II". *ACM Transactions on Modeling and Computer Simulation* 4 (3): 254–266.

Matsumoto, M., and Y. Kurita. 1996. "Strong Deviations from Randomness in m-Sequences Based on Trinomials". *ACM Transactions on Modeling and Computer Simulation* 6 (2): 99–106.

Matsumoto, M., and T. Nishimura. 1998. "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator". *ACM Trans. on Modeling and Computer Simulation* 8 (1): 3–30.

J. Maurer and S. Watanabe 2017. "Boost Random Number Library". http://www.boost.org/doc/libs/1_64_0/doc/html/boost_random/reference.html.

Metropolis, N., G. Reitwiesner, and J. von Neumann. 1950. "Statistical Treatment of Values of First 2000 Decimals Digits of e and π Calculated on the ENIAC". *Mathematical Tables and Other Aids to Computation* 4.

Metropolis, N. C. 1956. "Phase Shifts – Middle-Squares – Wave Equation". In *Symposium on Monte Carlo Methods*, edited by H. A. Meyer, 29–36: John Wiley & Sons.

Mood, A. M. 1940. "The Distribution Theory of Runs". *The Annals of Mathematical Statistics* 11 (4): 367–392.

Nance, R. E., and C. Overstreet, Jr. 1978. "Some Experimental Observations on the Behavior of Composite Random Number Generators". *Operations Research* 26 (5): 915–935.

Niederreiter, H. 1978. "Quasi-Monte Carlo Methods and Pseudorandom Numbers". *Bulletin of the American Mathematical Society* 84 (6): 957–1041.

Niederreiter, H. 1986. "A Pseudorandom Vector Generator Based on Finite Field Arithmetic". *Mathematica Japonica* 31:759–774.

Niederreiter, H. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*, Volume 63 of *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*. Philadelphia, PA: SIAM.

Niederreiter, H., and I. E. Shparlinski. 2002. "Recent Advances in the Theory of Nonlinear Pseudorandom Number Generators". In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, edited by K.-T. Fang, F. J. Hickernell, and H. Niederreiter, 86–102. Berlin: Springer-Verlag.

Nishimura, T. 2000. "Tables of 64-bit Mersenne Twisters". *ACM Transactions on Modeling and Computer Simulation* 10 (4): 348–357.

NIST 2001. "ADVANCED ENCRYPTION STANDARD (AES)". FIPS-197, U.S. DoC/National Institute of Standards and Technology. See http://csrc.nist.gov/CryptoToolkit/tkencryption.html.

Ore, O. 1948. *Number Theory and its History*. New York: McGraw-Hill.

Panneton, F., and P. L'Ecuyer. 2005. "On the Xorshift Random Number Generators". *ACM Transactions on Modeling and Computer Simulation* 15 (4): 346–361.

Panneton, F., P. L'Ecuyer, and M. Matsumoto. 2006. "Improved Long-Period Generators Based on Linear Recurrences Modulo 2". *ACM Transactions on Mathematical Software* 32 (1): 1–16.

Park, S. K., and K. W. Miller. 1988. "Random Number Generators: Good Ones Are Hard to Find". *Communications of the ACM* 31 (10): 1192–1201.

Pathria, R. K. 1962. "A Statistical Study of Randomness Among the First 10,000 Digits of π". *Mathematics of Computation* 16:188–197.

Peres, Y. 1992. "Iterating Von Neumann's Procedure for Extracting Random Bits". *Annals of Statistics* 20 (1): 590–597.

R Core Team 2015. *R: A Language and Environment for Statistical Computing, Reference Index*. Vienna, Austria: R Foundation for Statistical Computing.

RAND Corporation 1955. *A Million Random Digits with 100,000 Normal Deviates*. The RAND Corporation.

Read, T. R. C., and N. A. C. Cressie. 1988. *Goodness-of-Fit Statistics for Discrete Multivariate Data*. Springer Series in Statistics. New York, NY: Springer-Verlag.

Ripley, B. D., and B. W. Silverman. 1978. "Quick Tests for Spatial Interaction". *Biometrika* 65 (3): 641–642.

Rotenberg, A. 1960. "A New Pseudo-Random Number Generator". *Journal of the ACM* 7 (1): 75–77.

Rukhin, A., J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. 2001. "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications". NIST special publication 800-22, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA. See http://csrc.nist.gov/rng/.

Salmon, J. K., M. A. Moraes, R. O. Dror, and D. E. Shaw. 2011. "Parallel random numbers: as easy as 1, 2, 3". In *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 16:1–16:12. New York: ACM.

Shchur, L. N., J. R. Heringa, and H. W. J. Blöte. 1997. "Simulation of a Directed Random-Walk Model: The Effect of Pseudo-Random Number Correlations". *Physica A* 241:579–592.

Sowey, E. R. 1972. "A Chronological and Classified Bibliography on Random Number Generation and Testing". *International Statistical Review / Revue Internationale de Statistique* 40 (3): 355–371.

Stipčević, M., and Ç. K. Koç. 2014. "True Random Number Generators". In *Open Problems in Mathematics and Computational Science*, edited by Ç. K. Koç, 275–315. Springer International Publishing.

Sýs, M., Z. Říha, and V. Matyáš. 2016. "Algorithm 970: Optimizing the NIST Statistical Test Suite and the Berlekamp-Massey Algorithm". *ACM Transactions on Mathematical Software* 43 (3): 27:1–27:11.

Taussky, O., and J. Todd. 1954. "Generation and testing of pseudo-random numbers". Technical report, National Bureau of Standards. report number 3370.

Tausworthe, R. C. 1965. "Random Numbers Generated by Linear Recurrence Modulo Two". *Mathematics of Computation* 19:201–209.

Tezuka, S. 1995. *Uniform Random Numbers: Theory and Practice*. Norwell, MA: Kluwer Academic Publishers.

Tezuka, S., and P. L'Ecuyer. 1991. "Efficient and Portable Combined Tausworthe Random Number Generators". *ACM Transactions on Modeling and Computer Simulation* 1 (2): 99–112.

Tezuka, S., P. L'Ecuyer, and R. Couture. 1993. "On the Add-with-Carry and Subtract-with-Borrow Random Number Generators". *ACM Transactions of Modeling and Computer Simulation* 3 (4): 315–331.

Thomson, W. E. 1959. "ERNIE–A Mathematical and Statistical Analysis". *Journal of the Royal Statistical Society* A122:301–324.

Tippett, L. H. C. 1927. *Random Sampling Numbers*. Cambridge University Press.

Tootill, J. P. R., W. D. Robinson, and A. G. Adams. 1971. "The Runs Up-and-Down Performance of Tausworthe Pseudo-Random Number Generators". *Journal of the ACM* 18:381–399.

Tootill, J. P. R., W. D. Robinson, and D. J. Eagle. 1973. "An Asymptotically Random Tausworthe Sequence". *Journal of the ACM* 20:469–481.

Vattulainen, I., T. Ala-Nissila, and K. Kankaala. 1994, 11. "Physical Tests for Random Numbers in Simulations". *Physical Review Letters* 73 (19): 2513–2516.

Vigna, S. 2016. "An Experimental Exploration of Marsaglia's Xorshift Generators, Scrambled". *ACM Transactions on Mathematical Software* 42 (4): 30:1–30:23.

von Neumann, J. 1951. "Various techniques used in connection with random digits". In *The Monte Carlo Method*, edited by A. S. Householder et al., Volume 12, 36–38. National Bureau of Standards, Applied Mathematics Series.

Wang, D., and A. Compagner. 1993. "On the Use of Reducible Polynomials as Random Number Generators". *Mathematics of Computation* 60:363–374.

Weyl, H. 1916. "Über die Gleichverteilung von Zahlen mod. Eins". *Math. Ann.* 77:313–352.

Wichmann, B. A., and I. D. Hill. 1982. "An Efficient and Portable Pseudo-random Number Generator". *Applied Statistics* 31:188–190. See also corrections and remarks in the same journal by Wichmann and Hill, **33** (1984) 123; McLeod **34** (1985) 198–200; Zeisel **35** (1986) 89.

Wikipedia 2017. "Comparison of hardware random number generators — Wikipedia, The Free Encyclopedia". [Online; accessed 30-March-2017].

Wolfowitz, J. 1944. "Asymptotic Distribution of Runs Up and Down". *The Annals of Mathematical Statistics* 15 (2): 163–172.

Yee, A. J. 2017. "Y-Cruncher—A Multi-Threaded Pi-Program".

Zeisel, H. 1986. "A Remark on Algorithm AS 183.". *Applied Statistics* 35:89.

Zierler, N. 1959. "Linear Recurring Sequences". *Journal of the Society for Industrial and Applied Mathematics* 7 (1): 31–48.

AUTHOR BIOGRAPHY

PIERRE L'ECUYER is Professor in the Département d'Informatique et de Recherche Opérationnelle, at the Université de Montréal, Canada. He holds the Canada Research Chair in Stochastic Simulation and Optimization and an Inria International Chair in Rennes, France. He is a member of the CIRRELT and GERAD research centers. His main research interests are random number generation, quasi-Monte Carlo methods, efficiency improvement via variance reduction, sensitivity analysis and optimization of discrete-event stochastic systems, and discrete-event simulation in general. He has published over 260 scientific articles, book chapters, and books, and has developed software libraries and systems for random number generation and stochastic simulation (SSJ, TestU01, RngStreams, Lattice Builder, etc.). His Google Scholar H-index is 57, with an I10 index of 166. He has been a referee for 142 different scientific journals. He has served as Editor-in-Chief for *ACM Transactions on Modeling and Computer Simulation* from 2010 to 2013. He is currently Associate Editor for *ACM Transactions on Mathematical Software*, *Statistics and Computing*, and *International Transactions in Operational Research*. More information can be found on his web page: http://www.iro.umontreal.ca/~lecuyer.