Four-tap shift-register-sequence random-number generators

Robert M. Ziff

Citation: Computers in Physics **12**, 385 (1998); doi: 10.1063/1.168692
View online: https://doi.org/10.1063/1.168692
View Table of Contents: https://aip.scitation.org/toc/cip/12/4
Published by the American Institute of Physics

Four-tap shift-register-sequence random-number generators

Robert M. Ziff[a)]
Department of Chemical Engineering, University of Michigan, Ann Arbor, Michigan 48109-2136

(Received 16 October 1997; accepted 22 April 1998)

Correlations in the generalized feedback shift-register random-number generator are shown to be greatly reduced when the number of feedback taps is increased from two to four (or more) and the tap offsets are made large. Simple formulas for producing maximal-cycle four-tap rules from available primitive trinomials are given, and explicit three- and four-point correlations are found for some of those rules. Several generators are also tested using a simple but sensitive random-walk simulation that relates to a problem in percolation theory. While virtually all two-tap generators fail this test, four-tap generators with offsets greater than about 500 pass it, have passed tests carried out by others, and appear to be good multipurpose high-quality random-number generators. © *1998 American Institute of Physics.* [S0894-1866(98)01704-0]

INTRODUCTION

The ability to efficiently generate random (or ''pseudorandom'') sequences of numbers with good statistical properties is crucial to many problems in computational physics. Indeed, as ever faster computers allow ever more precise calculations to be carried out, the demands on the quality of the generator keeps increasing. Various schemes to generate random numbers have been developed and extensively studied over the years—see, for example, the reviews in Refs. 1–4. Yet, in spite of decades of work, the quality of random-number generators remains an issue. In this article we discuss one type of popular generator, the generalized feedback shift-register (GFSR) generator, and show that under certain circumstances it can yield sequences of very high quality.

The GFSR generator $R(a,b,c,\dots)$ produces pseudorandom numbers by the linear recursion[5–7]

$$x_n = x_{n-a} \oplus x_{n-b} \oplus x_{n-c} \oplus \dots, \qquad (1)$$

where $\oplus$ is the exclusive-or operation (addition modulo 2) and $a,b,c,\dots$ are the feedback taps. Here the x_n are either single bits or multibit words, in which case the $\oplus$ operation is carried out bitwise. This recursion was first studied extensively by Golomb[6] in the context of computer science, where it has many other applications, including cryptology and error-correcting codes. Its use as a random-number generator was introduced to the computational physics community by Kirkpatrick and Stoll,[8] who suggested the two-tap rule R(103,250), and it became fairly popular due to its simplicity and generally accepted quality.

However, it is now widely known that such generators, in particular those with two-tap rules such as R(103,250), have serious flaws. Many years ago, Hoogland *et al.*[9] reported irregularities in an Ising-model simulation using R(15,127). Parisi and Rapuano[10] showed errors in the results from a similar generator, R(97,127). The present author found problems using R(103,250) in a hull-walk simulation[11] and experimented with an empirical combination generator.[12] Marsaglia[2] demonstrated very poor behavior with R(24,55) and smaller generators, and advised against using generators of this type altogether.

More recently, Ferrenberg *et al.*[13] found that R(103,250) leads to results more than 100 standard deviations from the true values in Ising-model simulations utilizing the Wolff cluster-flipping Monte-Carlo algorithm. Coddington[14] confirmed this observation with an extensive study involving a large number of various random-number generators. Grassberger found striking errors in an efficient depth-first self-avoiding random-walk algorithm when R(103,250) was used.[15] Vattulainen *et al.*[16] devised a number of simple tests that clearly show the effective correlations and deficiencies in two-tap GFSR generators. And very recently, Shchur *et al.*[17] simplified the one-dimensional Wolff algorithm to a repeating one-dimensional random-walk test, which they found to clearly fail when R(103,250) is used.

The basic problem of two-tap generators $R(a,b)$ is that they have a built-in three-point correlation between x_n, x_{n-a}, and x_{n-b}, simply given by the generator itself, such that if any two of the x_n are known, the third follows directly from the recursion $x_n = x_{n-a} \oplus x_{n-b}$. These correlations extend over the size $p = \max(a,b)$ of the generator and can lead to significant errors, as shown for example in a recent study of Schmid and Wilding.[18]

Other problems with this generator are also known. Compagner and Hoogland[19] have shown how a pattern of all 1's in the initialization string leads to complex patterns of subsequent bits that persists for an unexpectedly long time. Shchur *et al.*[17] pointed out that for an event that occurs with a probability close to 1 (such as 31/32), it is not too unlikely for 249 successive true outcomes to occur, which then leads to a serious error at the 250th step when the R(103,250) generator is used.

For reasons like these, many people have, over the

[a)]E-mail: rziff@umich.edu

years, advocated using larger tap offset values $a,b,\ldots$ and increasing the number of those taps from two to four or more (they are always even in number for maximal-cycle generators[6]). Compagner and co-workers have considered generators with offsets as large as 132,049,[20] and have proposed combining two generators to effectively make multitap rules.[21,22] Heuer *et al.*[23] have also discussed similar combination generators recently. The advantages of using larger offsets are well documented; for example, Ferrenberg *et al.*[13] found the generator R(216,1279) to be nearly acceptable for their problem, and Coddington[14] showed that R(1393,4423) reduces the error below the measurable limit for the simulation cutoff that he used. Similar trends were seen by Compagner and Hoogland[19] and by Vattulainen *et al.* [16]

However, the use of more than two taps has not been common in practice. One reason is undoubtedly that tables of primitive polynomials on GF(2) (the Galois field on binary numbers) of order higher than three, which are needed to construct maximal-cycle rules, have been limited (although some have appeared more recently[24,25]), and their direct determination is a nontrivial exercise in number theory. Golomb has given a prescription for making new generators from existing ones based upon sequence decimation,[26] which can be used to construct multitap rules. In this article, we simplify this procedure by giving explicit formulas for 3, 5, and 7 decimation of two-tap rules, in which cases four-tap rules always result. These four-tap rules generate, in single calls, the same sequences that come from D decimation of the corresponding two-tap generators.

It turns out that this decimation procedure has been frequently employed in a literal sense simply by using every Dth call of a given generator. For example, Ferrenberg *et al.*[13] used every fifth call of the generator R(103,250), and found that its problems apparently disappear. Below we show that this five-call process is equivalent to making a single call of the four-tap generator R(50,103,200,250), and also discuss the inherent four-point correlations that generator possesses. Coddington[14] and Vattulainen *et al.*[16] also utilized this explicit decimation procedure. From a speed point of view, however, it is clearly advantageous to use a four-tap rule instead of having to make multiple calls of a two-tap rule for each random number needed.

Recently some lists of higher-order primitive polynomials have appeared in the literature. Those of André *et al.*[27] concern relatively small offset values p and have insufficient cycle lengths. Note that these (and other) authors advocate using many more feedback taps—of the order of $p/2$—which, however, would be impractical for the large p recommended here. Some larger primitive pentanomials have been given by Kurita and Matsumoto[24] and more recently by Živković,[25] but none of these has been studied in this work, which was carried out mainly during 1992–1994.

The formulas for constructing four-tap generators are given in Sec. I, along with proofs. In Sec. II, the correlations on smaller generators are found explicitly, and show that four-tap rules are vastly superior to two-tap rules in regards to three- and four-point correlations, except for certain classes of four-tap rules that have more prominent four-point correlations and therefore should be avoided. In Sec. III, a new test for random-number generators that makes use of a kinetic self-avoiding random walk related to percolation and the lattice Lorentz gas[28] is introduced. While the two-tap and smaller four-tap generators fail the test, four-tap generators with moderately large offsets pass, and suggest that with larger offsets, the error should be nearly unmeasurable. This test is evidently particularly sensitive to the type of asymmetric correlation that occurs with these generators. Some of these four-tap generators have also been tested on different problems by Coddington[14] and Vattulainen *et al.*,[16] who confirmed the trends seen here.

I. RULES FOR FOUR-TAP GENERATORS

The taps $(a,b,c,\ldots)$ are chosen so that the corresponding polynomial $1+z^a+z^b+z^c+\ldots$ is primitive over GF(2), guaranteeing that the cycle length will be the maximum possible value 2^p-1, where $p=\max(a,b,c,\ldots)$.[6,29] Besides giving the maximum possible number of random numbers before repeating, maximal rules have the advantage that they can be initialized with any sequence (other than all zeros). For two-tap rules, values of a and b can be found from extensive tables of primitive trinomials.[20,30,31] Higher-order polynomials can be generated from these trinomials by using a formal procedure based upon the concept of decimation.[6]

In D decimation, every Dth term of a given sequence is selected to produce a new sequence. The resulting sequence also satisfies a recursion like (1), corresponding to a polynomial of order p, although the number of taps is in general different. For some special cases, we have found simple formulas that give four-tap rules directly. Before deriving them, we first introduce the following alternate notation for the recursion (1): Let $[a,b,c,\ldots]$ indicate that the x_n satisfy the relation

$$x_{n-a}\oplus x_{n-b}\oplus x_{n-c}\oplus\ldots=0 \qquad (2)$$

for all n. Thus, $[0,a,b]$ is an equivalent way to write R (a,b). These relations satisfy some obvious properties: If $[a,b,c\ldots]$ is satisfied on a given sequence, then $[a+k,b+k,c+k\ldots]$ will also be satisfied for any k on that sequence (shift operation). Furthermore, if both $[a,b,\ldots]$ and $[a',b',\ldots]$ are satisfied, then their union or sum $[a,a',b,b',\ldots]$ will also be satisfied (addition property). Finally, if an offset occurs twice in the list, then it can be eliminated, because $x_i\oplus x_i=0$: $[a,b,b,c,\ldots]=[a,c,\ldots]$.

When a shift-register sequence is decimated by any power of 2, the original sequence is reproduced exactly, only shifted.[6] To prove this, let us consider the sequence generated by $\mathrm{R}(a,b)=[0,a,b]$. By the shift property, $[a,2a,a+b]$ and $[b,a+b,2b]$ are also satisfied on this sequence. Adding these three relations together yields $[0,a,a,b,b,a+b,a+b,2a,2b]=[0,2a,2b]$, which implies that *every other* term in the original sequence satisfies $[0,a,b]$. Thus, it follows that the original sequence and the 2-decimated sequence are identical. Because the decimation wraps around the entire sequence, which is odd in length, the decimated sequence is of the same maximal length as the original sequence. This proof can be easily generalized for any (even) number of taps, and decimation by any power of 2.

When decimation is done by a number D that is not a power of 2, however, a new sequence representing a different rule will, in general, be produced. While in general the resulting number of taps varies and may be large, it turns out that four-tap rules always result when a two-tap rule R (a,b) is decimated by $D=3$, 5, and 7. Those four-tap rules are given explicitly by the following formulas:

$$\mathrm{R}(a,b)\times 3 = \begin{cases} \mathrm{R}(a,a/3,2a/3,b) & 3|a, \quad (3a) \\ \mathrm{R}(a,(2a+b)/3,(a+2b)/3,b) & 3|(a-b), \quad (3b) \end{cases}$$

$$\mathrm{R}(a,b)\times 5 = \begin{cases} \mathrm{R}(a,a/5,4a/5,b) & 5|a, \quad (4a) \\ \mathrm{R}(a,(4a+b)/5,(a+4b)/5,b) & 5|(a-b), \quad (4b) \\ \mathrm{R}(a,(a+b)/5,2(a+b)/5,b) & 5|(a+b), \quad (4c) \\ \mathrm{R}(a,(3a+b)/5,(a+2b)/5,b) & 5|(2a-b), \quad (4d) \end{cases}$$

$$\mathrm{R}(a,b)\times 7 = \begin{cases} \mathrm{R}(a,(a+b)/7,3(a+b)/7,b) & 7|(a+b), \quad (5a) \\ \mathrm{R}(a,(5a+b)/7,(a+3b)/7,b) & 7|(2a-b), \quad (5b) \end{cases}$$

where $D|a$ indicates that a is divisible by D (D divides a). The remaining cases follow by switching a and b in the various formulas—for example, when $2b-a$ is divisible by D, then a and b must be switched in (4d) and (5b). Formulas for $7|a$ and $7|(a-b)$ are not listed because these cases do not exist for primitive trinomials.

We deduced these decimation formulas by examining specific examples using both Golomb's methods[6,26] and a numerical search procedure. We then verified the formulas by applying the shift and add properties given above.

For example, consider the case of (3a). Shifting $[0,a,b]$ by b, $2a$, $2b$ and $a+b$, respectively, yields the following five relations:

$[0,a,b]$	original rule,
$[b,a+b,2b]$	original rule shifted by b,
$[2a,3a,2a+b]$	original rule shifted by $2a$,
$[2b,2b+a,3b]$	original rule shifted by $2b$,
$[a+b,2a+b,2b+a]$	original rule shifted by $a+b$.

Summing these and canceling out common terms yields

$$[0,a,2a,3a,3b].$$

The final relationship (a five-point correlation) holds for any rule $\mathrm{R}(a,b)$. However, when a is divisible by 3, then all five elements are divisible by 3, so it follows that the 3-decimated sequence satisfies $[0,a/3,2a/3,a,b]$ or the rule $\mathrm{R}(a/3,2a/3,a,b)$ as given in (3a).

Likewise, for (3b), sum

$[0,2a,2b]$,	2-decimated rule,
$[2a,3a,2a+b]$,	original rule shifted by $2a$,
$[2b,2b+a,3b]$,	original rule shifted by $2b$,

to find

$$[0,3a,2a+b,a+2b,3b],$$

which implies (3b) when $2a+b$ and $a+2b$ are both divisible by 3, which occurs when $a-b$ is divisible by 3. For primitive trinomials, it is always true that either a, b, or $a-b$ is divisible by 3,[6] so (3) contains all cases. Proofs for 5 and 7 decimation are similar. Note that decimations by more than 7 (and not a power of 2) do not, in general, give four-tap rules but, rather, ones having many more taps. In this regard, $D=3$, 5, and 7 appear to be special cases.

Using the above formulas with a and b taken from existing tables of primitive trinomials,[20,31] we can find numerous four-tap generators. However, some of these generators will not be of maximal-cycle length. In order that the cycle of the decimated sequence be the same as that of the original sequence, it is necessary that D and 2^p-1 have no common divisors. This requirement is satisfied for 3 decimation when $p \bmod 2\neq 0$, for 5 decimation when $p \bmod 4\neq 0$, and for 7 decimation when $p \bmod 3\neq 0$. (On the other hand, when these conditions are not satisfied, the cycle length is less than the maximum by a factor of 3, 5, or 7 and is therefore still enormous when p is large, so this criterion may not be that significant.) An additional criterion for selecting rules to decimate, dealing with four-point correlations, will be discussed below.

II. CORRELATIONS

The relation $[a,b,c,\dots]$ represents a correlation among the points x_{n-a}, x_{n-b}, x_{n-c}, ... These are very strong correlations; for example, $[0,a,b]$, implies that if any two of x_n, x_{n-a}, and x_{n-b} are known, then the third is completely determined, as mentioned above. The sequences generated by (1) are literally laced with such correlations. First of all, the basic correlation is given by the defining rule itself, $\mathrm{R}(a,b,c,\dots)$, in that $[0,a,b,c,\dots]$ is satisfied for each n. Then there is also a whole spectrum of three-point correlations in the system: By the so-called ''cycle and add'' property,[6,19] there exists an s such that $[0,r,s]$ is satisfied for each value of $r=1,2,3,\dots 2^p-1$. The value of $\max(r,s)$ is typically on the order of $2^{p/2}$ to 2^p when the defining rule is a pentanomial or higher. However, when the defining rule is a trinomial $\mathrm{R}(a,b)$, s will be of the order p for $r=a$, b, $2a$, $2b$, $4a$, $4b$, etc. These closely spaced three-point correlations interact to form numerous closely spaced four-point, five-point, and higher correlations, which are unavoidable when two-tap rules are chosen.

For most applications, correlations in a random-number sequence involving the fewest number of points will undoubtedly be the most serious. For example, if a kinetic random walk returns to the same region in space at steps n, $n-a$ and $n-b$ for some n, then its behavior would be strongly affected by the three-point correlation $[0,a,b]$ in the random-number sequence. Higher correlations would correspond to more coincidences in the motion of the walk and should therefore be less likely. It is thus reasonable to assume that the reduction of three-point correlations is most important, followed by four-point correlations, and so on.

Using a four-tap rule $\mathrm{R}(a,b,c,d)$ immediately eliminates the overriding three-point correlation $[0,a,b]$ inherent in a two-tap rule $\mathrm{R}(a,b)$, and the remaining three-point correlations (which still exist) are widely spaced as mentioned above. The four-point correlations of a four-tap rule are also generally widely spaced. An exception occurs when the four-tap rule follows from a D decimation of a two-tap rule $\mathrm{R}(a,b)$ and a, b, or $a-b$ is divisible by D. In this case, the correlation offsets are small and can be derived explicitly. For example, the 3 decimation of $\mathrm{R}(a,b)$ yields $[0,a/3,2a/3,a,b]$ according to (3a). By shifting this five-point correlation by $a/3$ and adding, one finds the four-point correlation

$$\begin{aligned}&[0,a/3,2a/3,a,b]+[a/3,2a/3,a,4a/3,a/3+b]\\&\quad=[0,4a/3,b,a/3+b].\end{aligned}\tag{6}$$

The spread of this correlation is of the order of p, not 2^p. Such a four-point correlation in $\mathrm{R}(38,89)\times 3=\mathrm{R}(38,55,72,89)$ (where $89-38$ is divisible by 3) was observed in Ref. 17. A similar result holds for the 5-decimation rules (4a) and (4b). To avoid these relatively closely spaced four-point correlations, *all* 3 decimations (3), and the 5 decimations of cases (4a) and (4b), should not be used. These cases will not be investigated below, except for the rule $\mathrm{R}(103,250)\times 5=\mathrm{R}(50,103,200,250)$ which was considered in Ref. 13. Here, 250 is divisible by 5, and as a consequence this rule possesses the relatively closely spaced four-point correlation [0,309,359,800].

For generators produced by other rules, it appears that the correlations can only be found by a search procedure, in which a long sequence of bits is generated, and possible correlations are checked exhaustively. To make this procedure feasible for larger p, we made a list of up to 2^{21} 32-bit subsequences and sorted them with keys pointing to their location in the sequence, which made it possible to quickly find if a subsequence generated by a trial correlation occurs in the original sequence. Details will be presented elsewhere. This procedure turned out to be practical for finding three- and four-point correlations for p up to about 50, which would be impossible by a simple brute-force search.

Some representative results from this search are given below. Each line shows, respectively, the way the rule was generated from the two-tap rules of Ref. 31, the equivalent four-tap rule from (4) or (5) (which also represents the smallest five-point correlation $[0,a,b,c,d]$ in the sequence), and the smallest four- and three-point correlations found by the search. These results are

$$\mathrm{R}(5,17)\times 7=\mathrm{R}(5,6,8,17))=[0,77,79,101]=[0,67,83],\tag{7a}$$

$$\begin{aligned}\mathrm{R}(5,23)\times 7&=\mathrm{R}(4,5,12,23)\\&=[0,13,50,421]\\&=[0,1153,4933],\end{aligned}\tag{7b}$$

$$\begin{aligned}\mathrm{R}(3,31)\times 5&=\mathrm{R}(3,8,13,31)\\&=[0,87,199,397]\\&=[0,30\,189,34\,284],\end{aligned}\tag{7c}$$

$$\begin{aligned}\mathrm{R}(6,31)\times 7&=\mathrm{R}(6,7,23,31)\\&=[0,40,623,2216]\\&=[0,14\,487,101\,088],\end{aligned}\tag{7d}$$

$$\begin{aligned}\mathrm{R}(8,39)\times 7&=\mathrm{R}(8,9,29,39)\\&=[0,111,1072,7006]\\&=[0,172\,074,758\,257],\end{aligned}\tag{7e}$$

$$\begin{aligned}\mathrm{R}(3,41)\times 7&=\mathrm{R}(3,8,18,41)\\&=[0,4280,6131,8713]\\&=[0,351\,102,1\,716\,109],\end{aligned}\tag{7f}$$

$$\begin{aligned}\mathrm{R}(20,47)\times 7&=\mathrm{R}(20,21,23,47)\\&=[0,33\,579,138\,448,150\,900]\\&=[0,8\,474\,125,11\,136\,544],\end{aligned}\tag{7g}$$

$$\begin{aligned}\mathrm{R}(21,47)\times 5&=\mathrm{R}(21,22,23,47)\\&=[0,63\,608,148\,485,156\,350]\\&=[0,11\,941\,097,13\,215\,912].\end{aligned}\tag{7h}$$

Thus, for example, the four-tap rule R(5,6,8,17) generates a series that has the three-point correlation [0,67,83], four-point correlation [0,77,79,101], as well as the inherent five-point correlation [0,5,6,8,17]. Note that the two-tap rule R(67,83) corresponding to this three-tap correlation can only be used to generate the sequence produced by R (5,6,8,17) if it is started up correctly with the 83 bits from the latter's sequence, because the sequence generated by R (5,6,8,17) is only one of many cycles of the nonmaximal rule R(67,83). Therefore, the correlations in brackets, such as [0,67,83], should not be interpreted as suggested rules for random-number generators.

The above results clearly show that the separation in the three-and four-point correlations increases rapidly as p increases. In fact, the extent of the smallest three-point correlation appears to grow roughly as $2^{p/2}$, and the extent of the smallest four-point correlations as $2^{p/3}$. Clearly, for larger p, such correlations will be irrelevant, and the most important correlations in four-tap rules will be the five-point correlations generated by the rule itself.

Additional maximal length rules can be generated by Golomb's method of repeated 3 decimation.[26] (Indeed, for some cases of p, repeated 3 decimation of a single maximal-length rule yields the complete cycle of all possible maximal-length rules.) For comparison, we have studied the behavior of some of these other rules. We found that, for a given p, the three- and four-point correlations are roughly equivalent to the correlations found in simple 5- and 7-decimation rules. For example, for the four-tap rule R(23,27,40,41), found by successively 3-decimating R(3,41) 107,005,025 times—equivalent to decimating once by $3^{107,005,025} \bmod (2^{41}-1)=1{,}962{,}142{,}349{,}662$—we find

$$\begin{aligned} R(3,41)\times 1{,}962{,}142{,}349{,}662 &= R(23,27,40,41) \\ &= [0,20\,573,22\,443,25\,575] \\ &= [0,429\,959,1\,013\,792], \end{aligned} \tag{8}$$

which may be compared with (7f). Six-tap rules with $p=41$ were found to possess similar three- and four-point correlations.

Some rules that follow from (4b), (4c) and (5) for larger offsets p are given by

$$R(38,89)\times 5 = R(33,38,61,89), \tag{9a}$$

$$R(11,218)\times 7 = R(11,39,95,218), \tag{9b}$$

$$R(216,1279)\times 5 = R(216,299,598,1279), \tag{9c}$$

$$R(216,1279)\times 7 = R(216,337,579,1279). \tag{9d}$$

$$R(471,9689)\times 5 = R(471,2032,4064,9689), \tag{9e}$$

$$R(471,9689)\times 7 = R(471,1586,6988,9689). \tag{9f}$$

$$R(33\,912,132\,049)\times 5 = R(33\,912,46\,757,59\,602,132\,049), \tag{9g}$$

$$R(33\,912,132\,049)\times 7 = R(33\,912,43\,087,61\,437,132\,049). \tag{9h}$$

The offset values of the three- and four-point correlations for these rules are greater than can be found by our search program and are presumably immense, judging by the trends seen in (7). To assess the quality of these generators, we turn to a test based upon a problem from percolation theory.

III. TEST ON THE RANDOM-WALK PROBLEM

The test we use is shown in Fig. 1. A walker starts at the lower left-hand corner of a square lattice, and heads in the diagonal direction toward the opposite corner. At each step it turns by 90° either clockwise or counterclockwise. When it encounters a site it had never visited before, the walker chooses which direction to turn with a 50–50 probability, while at a site that has been previously visited, it always turns so as not to retrace its path (a so-called kinetic self-avoiding trail on a square lattice). The boundary of the lattice is a square; the lower and left-hand sides are reflecting, while the upper and right-hand side are adsorbing. Clearly, by the perfect symmetry of the problem, the walker should first reach either the top or the right-hand sides with equal probability. We will see, however, that not all these random-number generators yield this simple result.

It turns out that this walk is precisely the kinetic self-

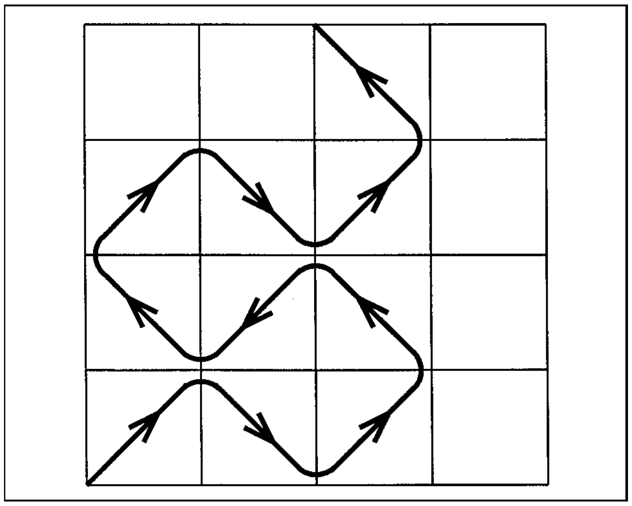

Figure 1. The random-walk algorithm used to test the random numbers shown for a system of size $L=4$. The test is whether the walker, which turns 90° to the left or right with equal probability at each newly visited site, first reaches the right or the top with equal probability. The left and bottom sides are reflecting.

avoiding walk that generates the hull of a bond percolation cluster at criticality. The lattice vertices visited by the walk are located at the centers of the bonds, and the two choices correspond to placing either a bond on the lattice or one on the dual lattice across that vertex point. The probability of reaching the upper side first corresponds to the crossing probability of a square, which is exactly 1/2, with no finite-size corrections for this particular system (bond percolation on a square lattice).[32–34] The walk is also identical to a lattice-Lorentz gas introduced by Ruijgrok and Cohen[28] with randomly oriented mirrors, to motion through a system of rotators as in the model of Gunn and Ortuño,[35] and to paths on the random tiling of Roux *et al.*[36] Note that this test is an actual algorithm that has been used in percolation studies;[34,37,38] it is not a ‘‘cooked-up'' problem designed specifically to reveal flaws in a specific random-number generator.

Using this procedure, we tested a variety of generators, including the two-tap generators R(11,218), R(103, 250), R(216,1279), R(576,3217), and R(471,9689), the four-tap generators R(20,27,34,41), R(3,26,40,41), R(1, 15,38,41), R(1,3,4,64), R(33,38,61,89), R(11,39,95,218), R(50,103,200,250), R(216,337,579,1279), and R(471, 1586,6988,9689), and the six-tap generators (determined through successive 3 decimation[26]) R(1,5,8,30,35,41), R(5,14,20,36,37,41), and R(18,36,37,71,89,124). Between 100,000 and 2,000,000 trials were simulated with each generator, implying a statistical error of about ± 0.001 in the crossing probability. The lattice was 4096×4096, and intermediate results for squares of side $L=64$, 128, 192, . . . ,4032 were also recorded. Figures 2 and 3 show the fraction of walks that first arrived at the upper boundary in each of these runs as a function of L. Clearly, some generators are very bad; for example, with the notorious R(103,250), the top of the 4096×4096 square was reached only 33% of the time! This error clearly cannot be statistical in origin; in fact, it is about 180 times the standard

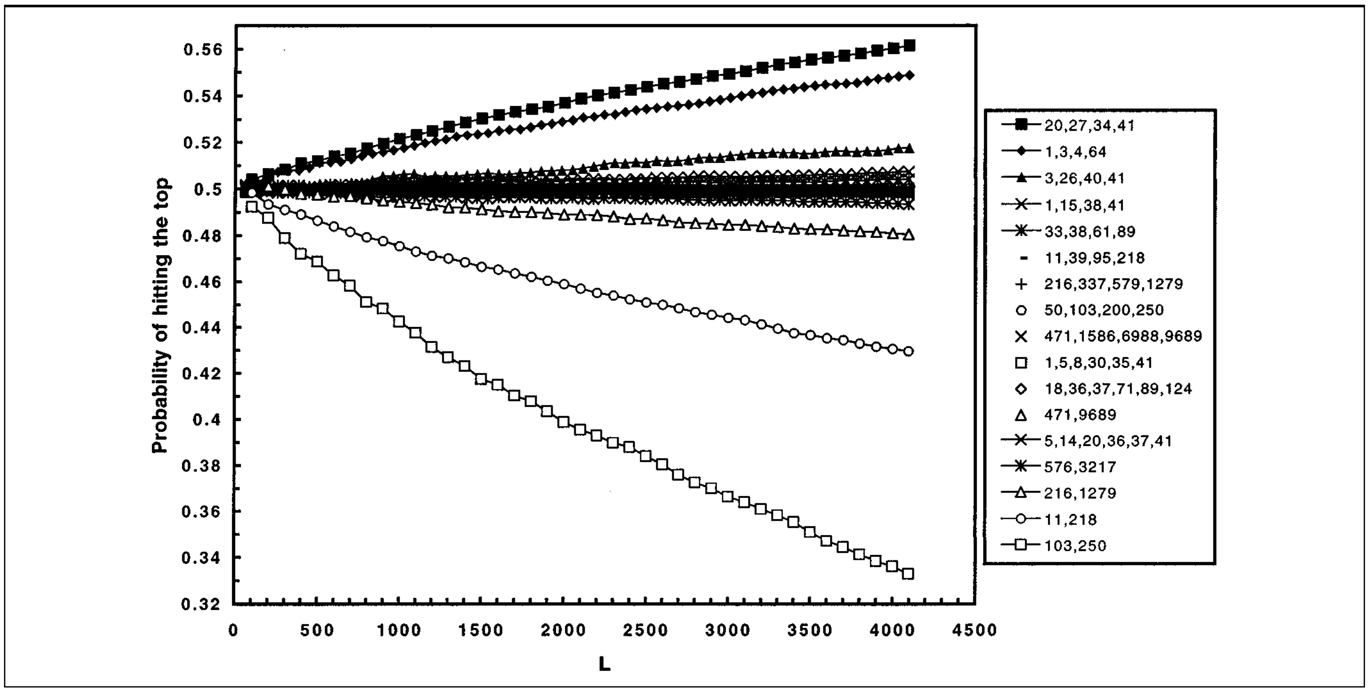

Figure 2. Plot of the probability of the walk reaching the top of an L×L system vs L, showing large deviations from the expected value of 1/2 for many of the generators.

deviation $\sigma = 0.001$. All of the smaller two-tap generators are poor, and even the largest one (with $p = 9689$) is barely within two standard deviations at $L = 4096$.

The four-tap generator with $p = 89$, on the other hand, begins to show deviations at large L, and the generator with $p = 218$ shows no visible deviations at all in this work. (However, in more recent tests of 10^8 runs on a lattice of size 256×256, we found some error creeping in for R(11,39,95,218), with the crossing probability at $L = 256$ given by $0.500\,30 \pm 0.000\,05$.[37]) Clearly, as p is increased, more random numbers need to be generated before the errors can be seen. For four-tap rules with p larger than about

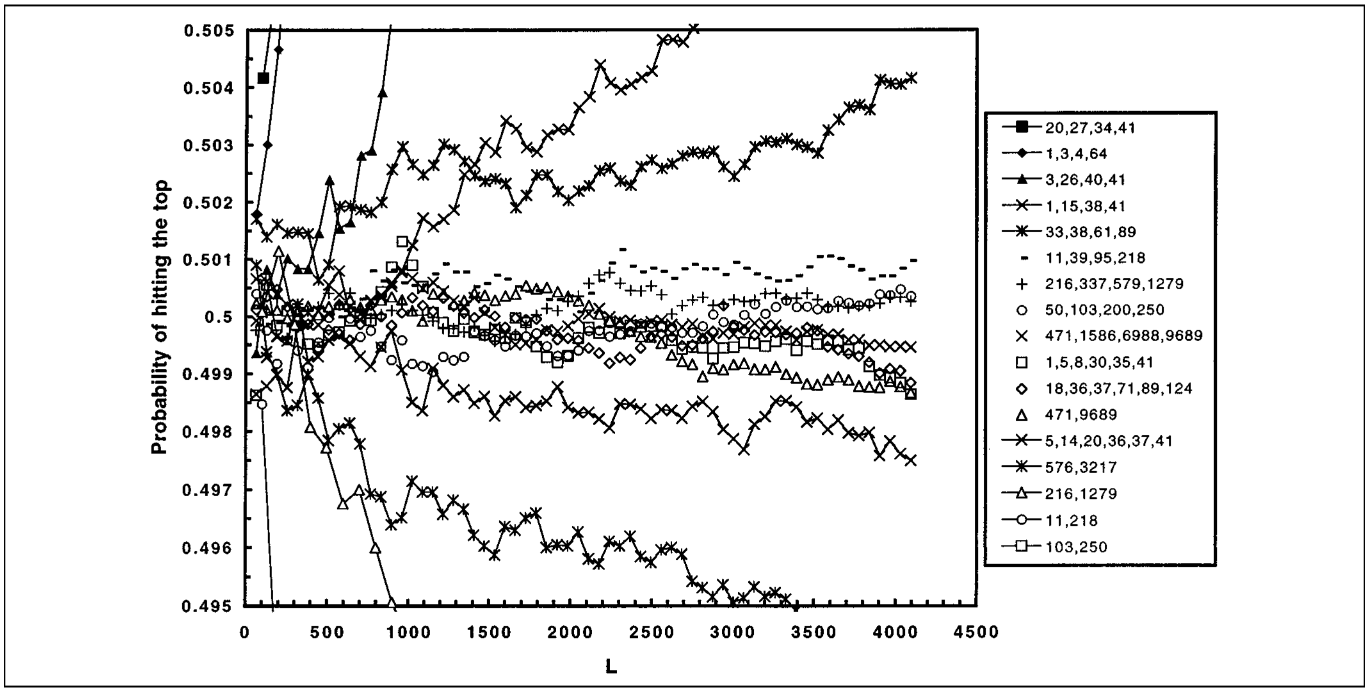

Figure 3. Central portion of Fig. 1 expanded vertically.

500, deviations in this test should be nearly impossible to uncover with present-day computers.

There are a number of interesting and puzzling aspects of these results. Evidently, two-tap generators give low results, four-tap generators give high results, and six-tap ones again give low ones. The supposedly bad generator R(50, 103,200,250), with its strong four-point correlations mentioned above, actually yields excellent results. Finally, the generators R(3,26,40,41) and R(1,15,38,41) are mirrors of each other, and so have identical (but mirrored) correlations of all points, and yet lead to noticeably different behavior. The explanation of these intriguing phenomena is a subject for future research. One might also investigate whether the choice to grow a new hull immediately after the previous one is completed, without any gap in the random-number sequence, has any bearing on the results.

Note that the plots in Figs. 2 and 3 are nearly, but not quite, linear. In fact, one can argue that the deviation must grow with a power of L that is less than or equal to 7/8. For, say the error is proportional to L^x with increasing L. This error will first be noticeable when the number of runs N_{runs} satisfies $N_{\text{runs}}^{-1/2} \sim L^x$ or $N_{\text{runs}} \sim L^{-2x}$. Because the path of the walk is a percolation hull, which has a fractal dimension of 7/4,[39] it follows that the number of random numbers generated per run grows as $L^{7/4}$. Thus, the total number of random numbers needed in order that the error can be observed grows as $\sim L^{7/4-2x}$. Now, the exponent in this expression cannot be negative, since that would imply that going to an infinite system would allow the error to be found with essentially no work. So we deduce $x \leq 7/8$. Numerically, a value of about $x = 0.7$ seems to give the best fit to the data in Fig. 2. That x is less than 7/8 implies that doing more runs on a smaller lattice, rather than fewer runs on a larger lattice, is actually a more efficient way to uncover the errors in these generators, to the extent that the power-law behavior L^x holds for small L.

Because this test is completely symmetric, the errors seen here highlight the fundamental asymmetry of the GFSR generator. Indeed, the basic exclusive-or operation has an inherent asymmetry to it, since two 0's or two 1's both result in a 0. For a correlation $[0,a,b]$, the three points x_n, x_{n-a}, and x_{n-b} can have only the values (0,0,0) and (0,1,1) (and permutations) that are clearly not symmetric in 1's and 0's. (This asymmetry is not in the total abundance of 0's and 1's, which are equally probable, but in their correlations.) Another way of demonstrating this asymmetry is to note that changing 1's to 0's and 0's to 1's in the initial seed sequence does not result in the complementary sequence being generated. Likewise, complementary subsequences are not equally likely.

Tests were also carried out using 31- and 48-bit linear congruence generators, and no errors were found. Evidently, congruence generators have a symmetry such that complementary subsequences are generated with equal probability, which guarantees that the probability of reaching the top is exactly 1/2. This result underscores the restriction that the test used here is not a definitive one, and does not uncover all errors in a random-number generator—as indeed no test does. It is necessary to use a battery of tests, as was done in Refs. 14 and 16. Both of those works included some of the four-tap generators we proposed, and both confirmed excellent behavior when p is large.

IV. CONCLUSIONS

Clearly, all GFSR random-number generators will eventually show detectable errors if a sufficiently long run is made. However, when the four-tap generators with p greater than about 500 is used, the amount of computer time that would be needed to uncover those errors would be prohibitive. The three- and four-point correlations of these generators are projected to be spread very far apart. Thus, such large four-tap generators appear useful as high-quality (pseudo-) random-number generators.

Two-tap generators, in contrast, do not pass the test carried out here, except perhaps those with the largest tap offsets. Thus, for critical applications, all two-tap generators, not just R(103,250), should be avoided. This includes all the two-tap generators discussed in Knuth's book,[1] for example.

If a problem is sensitive to the built-in five-point correlations of a four-tap generator, then a higher number of taps should be used. For this, the combination generators like those discussed by Compagner[21,22] and Heuer *et al.*[23] should be useful.

In spite of their known correlations as discussed here, there are many reasons that GFSR random-number generators remain of interest. In contrast to some combination generators, they are clean and well characterized; a large body of fundamental theory on their properties has been produced (i.e., in Ref. 4). Even with four taps, they remain fast and easy to program. Each bit is entirely independent, which is not the case for linear congruence generators or ''lagged-Fibonacci'' generators with addition or multiplication. Although they require storing a long list to obtain good behavior, the memory requirements are not a problem for present-day computers.

Over the last 10 years, we have carried out extensive simulations on a variety of problems in percolation and interacting particle models using the four-tap generators derived here. Our earlier work[40] made use of R(157,314,471, 9689), which derives from R(471,9689)×3; in more recent work [34,37,41,42] we switched to the 7-decimation generator (9f), because of the inherently strong four-point correlations in 3-decimation rules as discussed above—although in fact we never observed any problem with R(157,314, 471,9689), presumably because of its large offset $p = 9689$. In this work we often made checks with exact results when available, and never found any indication of bias. In a recent paper determining the bond percolation threshold for the Kagomé lattice,[41] we also checked the results of using R(471,1586,6988,9689) against runs using a 64-bit congruential generator, as well as the 3 decimation of R(471,1586,6988,9689) [thus equivalent to R(471, 9689)×21], and found complete consistency throughout.

In closing, we give an explicit example of the generator, written in a single line of the C programming language. It makes use of the define statement, which produces in-text substitution during the precompiling stage, so that no time is lost in a function call:

```
#define RandomInteger (++nd, ra[nd & M] = ra[(nd-A) & M] \
    ^ ra[(nd-B) & M] ^ ra[(nd-C) & M] ^ ra[(nd-D) & M])
```

The generator is called simply as follows

```
if(RandomInteger<prob)...
```

where, for rule (9f), for example, A=471, B=1586, C=6988, D=9689; and M=16383 (defined as constants), ra is an integer array over 0..M that is initialized using a standard congruential random number, nd is its index (an integer), & is the bitwise ''and'' operation, and ^ is the bitwise ''xor'' operation. ''Anding'' with M effectively causes the numbers to cycle endlessly around the list, when M+1 is chosen to be a power of 2 as above. The list in this example requires 64 kbytes of memory (16,384×4), if 32-bit (4-byte) integers are used. Here, prob is the probability of the event occurring, converted to an integer in the range of 0 to the maximum integer. A floating-point number can also be produced by dividing RandomInteger by the maximum integer (which depends upon the number of bits in ra), but this added step consumes additional time. Generating a random integer with this program takes about 50 ns on an HP 9000/780 workstation—a billion in less than a minute—and about 140 ns on a Macintosh 266-MHz G3 personal computer.

ACKNOWLEDGMENTS

The author gratefully acknowledges useful correspondence with H. Fredricksen and S. Golomb a number of years ago, and many helpful questions and comments from P. Grassberger in an extended electronic-mail interchange. He also thanks P. Coddington and I. Vattulainen, T. Ala-Nissila, and K. Kankaala for testing some of these four-tap generators and sharing their results with him prior to publication, as well as A. Compagner for sending him preprints of his work over the years. This material is based upon work supported by the U.S. National Science Foundation under Grant No. DMR-9520700.

REFERENCES

1. D. E. Knuth, *Seminumerical Algorithms, The Art of Computer Programming* (Addison–Wesley, Reading, MA, 1973), Vol. 2.
2. G. Marsaglia, in *Computer Science and Statistics: The Interface*, edited by L. Billard (Elsevier–North-Holland, Amsterdam, 1985).
3. P. L'Ecuyer, Ann. Oper. Res. **53**, 77 (1994).
4. H. Niederreiter, J. Comp. Appl. Math. **56**, 159 (1994); in *Monte-Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, edited by H. Niederreiter and P. J.-S. Shiue, Volume 106 of Lecture Notes in Statistics (Springer, Berlin, 1995).
5. R. C. Tausworthe, Math. Comput. **19**, 201 (1965).
6. S. W. Golomb, *Shift Register Sequences* (Holden Day, San Francisco, CA, 1967).
7. T. G. Lewis and W. H. Payne, J. Assoc. Comput. Mach. **20**, 456 (1973).
8. S. Kirkpatrick and E. P. Stoll, J. Comput. Phys. **40**, 517 (1981).
9. A. Hoogland, J. Spaa, B. Selman, and A. Compagner, J. Comput. Phys. **51**, 250 (1983).
10. G. Parisi and F. Rapuano, Phys. Lett. **157B**, 301 (1985).
11. R. M. Ziff, Phys. Rev. Lett. **56**, 545 (1986).
12. R. M. Ziff (unpublished). The work proposed alternating between R(103, 250) and R(30, 127) on the same list of numbers, which is presumably equivalent to using another multitap generator. However, it is not known whether maximal-cycle length is achieved in this method, and therefore the matter requires further study.
13. A. M. Ferrenberg, D. P. Landau, and Y. J. Wong, Phys. Rev. Lett. **69**, 3382 (1993).
14. P. D. Coddington, Int. J. Mod. Phys. C **5**, 547 (1994); **7**, 295 (1996).
15. P. Grassberger, Phys. Lett. A **181**, 43 (1993).
16. I. Vattulainen, T. Ala-Nissila, and K. Kankaala, Phys. Rev. Lett. **73**, 2513 (1994); Phys. Rev. E **52**, 3205 (1995).
17. L. N. Shchur, J. R. Heringa, and H. W. J. Blöte, Physica A **241**, 579 (1997); L. N. Shchur and H. W. J. Blöte, Phys. Rev. E **55**, R4905 (1997).
18. F. Schmid and N. B. Wilding, Int. J. Mod. Phys. C **6**, 781 (1995).
19. A. Compagner and A. Hoogland, J. Comput. Phys. **71**, 391 (1987).
20. J. R. Heringa, H. W. J. Blöte, and A. Compagner, Int. J. Mod. Phys. C **3**, 561 (1992).
21. A. Compagner, J. Stat. Phys. **63**, 883 (1991).
22. A. Compagner, Am. J. Phys. **59**, 700 (1991).
23. A. Heuer, B. Dünweg, and A. M. Ferrenberg, Comput. Phys. Commun. **103**, 1 (1997).
24. Y. Kurita and M. Matsumoto, Math. Comput. **56**, 194 (1991).
25. M. Živković, Math. Comput. **62**, 205 (1994).
26. S. W. Golomb, in *Combinatorial Mathematics and its Applications, Conference Proceedings*, edited by R. C. Bose and T. A. Dowling (University of North Carolina Press, Chapel Hill, 1969), pp. 358–370.
27. A. A. André, G. L. Mullen, H. Niederreiter, Math. Comput. **54**, 737 (1990).
28. Th. Ruijgrok and E. G. D. Cohen, Phys. Lett. A **133**, 415 (1988).
29. W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes* (MIT Press, Cambridge, MA, 1972).
30. N. Zierler and J. Brillhart, Inf. Control. **13**, 541 (1968); **14**, 566 (1969).
31. N. Zierler, Inf. Control. **15**, 67 (1969).
32. J. C. Cardy, J. Phys. A **25**, L201 (1992).
33. R. Langlands, P. Pouiliot, and Y. Saint-Aubin, Bull. Am. Math. Soc. **30**, 1 (1994).
34. R. M. Ziff, Phys. Rev. Lett. **69**, 2670 (1992).
35. J. M. F. Gunn and M. Ortuño, J. Phys. A **18**, L1095 (1985).
36. S. Roux, E. Guyon, and D. Sornett, J. Phys. A **21**, L475 (1988).
37. R. M. Ziff, Phys. Rev. E **54**, 2547 (1996).
38. J.-P. Hovi and A. Aharony, Phys. Rev. E **53**, 235 (1996).
39. B. Duplantier and H. Saleur, Phys. Rev. Lett. **59**, 539 (1987).
40. R. M. Ziff and G. Stell, University of Michigan Laboratory for Scientific Computing Report No. 4-88 (1988).
41. R. M. Ziff and P. Suding, J. Phys. A **30**, 5351 (1997).
42. C. D. Lorenz and R. M. Ziff, Phys. Rev. E **57**, 230 (1998).