The Annals of Applied Probability
1991, Vol. 1, No. 3, 462–480

A NEW CLASS OF RANDOM NUMBER GENERATORS[1]

BY GEORGE MARSAGLIA AND ARIF ZAMAN

The Florida State University

We introduce a new class of generators of two types: add-with-carry and subtract-with-borrow. Related to lagged-Fibonacci generators, the new class has interesting underlying theory, astonishingly long periods and provable uniformity for full sequences. Among several that we mention, we recommend particularly promising ones that will generate a sequence of 2^{1751} bits, or a sequence of 2^{1376} 32-bit integers, or a sequence of 2^{931} reals with 24-bit fractions—all using simple computer arithmetic (subtraction) and a few memory locations.

1. Introduction. We describe a new class of random number generators here. The most remarkable feature of these new generators is their ability to generate immensely long sequences, given an initial set of r seed values (computer words), with r typically from 20 to 50. Furthermore, with the exception of a negligible set, every possible r-tuple of words will appear in the sequence, a desirable uniformity feature that few generators share. Such immense periods have become more and more necessary with ultra-fast computers and exotic architectures, which might require thousands of streams of random numbers or have thousands of processors, each using parts of a single sequence so long that the probability of overlap is virtually zero.

This section briefly describes the most commonly used random number generators to provide the context for the new class of generators. The new generators can be grouped into four subclasses: two add-with-carry and two subtract-with-borrow generators. In Section 2 we define the two add-with-carry generators, and Section 3 describes the two subtract-with-borrow generators. Section 4 provides an analysis of the periods of these generators; then Section 5 describes the structure of the sequences that the new generators produce. Section 6 discusses the computational problems involved in finding parameters that lead to practical, long period implementations of the new generators and Section 7 gives an example of an add-with-carry generator suitable for expository or classroom use. A summary and some recommended implementations of the new generators are in Section 9.

Virtually all random number generators are based on theory which may be described as follows: We have a finite set X and a function f: $X \to X$ that takes elements of X into other elements of X. Given an initial (seed) value

Received October 1990; revised March 1991.
[1]Research supported by NSF Grant DMS-88-07976.
AMS 1980 *subject classifications*. 65C10, 10A30.
Key words and phrases. Random number generators, lagged-Fibonacci, add-with-carry, subtract-with-borrow, Monte Carlo.

$x \in X$, the generated sequence is

$$x,\ f(x),\ f^2(x),\ f^3(x),\ldots,$$

where $f^2(x)$ means $f(f(x))$, $f^3(x)$ means $f(f^2(x)) = f(f(f(x)))$ and so on. The three most common classes of random number generators are (1) congruential, (2) shift-register and (3) lagged-Fibonacci.

For *congruential generators*, the finite set X is the set of reduced residues of some modulus m and $f(x) = ax + b \bmod m$. Thus, with an initial element $x_0 \in X$, the generated sequence is

$$x_0, x_1, x_2, \ldots \text{ with } x_{n+1} = ax_n + b \bmod m.$$

A wide variety of choices for a, b and m have been described in the literature; see, particularly, Knuth [2] or Marsaglia [4] for methods for finding periods and establishing structure of congruential sequences.

For *shift-register generators*, the finite set X is the set of $1 \times k$ binary vectors $x = (b_1, b_2, \ldots, b_k)$ and the function f is a linear transformation, $f(x) = xT$, with T a $k \times k$ binary matrix and all arithmetic mod 2. With an initial binary vector x, the sequence is

$$x, xT, xT^2, xT^3, \ldots$$

with the matrix T chosen so that the period is long and multiplication by T is reasonably fast in computer implementation. See Marsaglia and Tsay [7] for methods for finding periods and establishing structure of shift-register sequences.

For *lagged-Fibonacci generators*, the finite set X is the set of $1 \times r$ vectors $x = (x_1, x_2, \ldots, x_r)$ with elements x_i in some finite set S on which there is a binary operation $\diamond$. The function f is defined by

$$f(x_1, x_2, \ldots, x_r) = (x_2, x_3, x_4, \ldots, x_r, x_1 \diamond x_{r+1-s}).$$

Informally, we describe a lagged-Fibonacci sequence as a set of r seed values followed by the rule for generating succeeding values:

$$x_1, x_2, \ldots, x_r, x_{r+1}, \ldots \text{ with } x_n = x_{n-r} \diamond x_{n-s},$$

but to formally define and establish the period and structure of such sequences we view them as iterates x, $f(x)$, $f^2(x), \ldots$ on the set X of $1 \times r$ vectors with elements in the set S on which the binary operation $\diamond$ is defined.

Various choices for S and $\diamond$ lead to interesting sequences: for example, when S is the set of reduced residues of some modulus m and $\diamond$ is addition or subtraction mod m; S is the set of reduced residues relatively prime to m and $\diamond$ is multiplication; S is the set of $1 \times k$ binary vectors and $\diamond$ is addition of binary vectors (exclusive-or); S is the set of floating-point computer numbers $0 \le x < 1$ having 24-bit fractions and $x \diamond y = \{\text{if } x > y \text{ then } x - y \text{ else } x - y + 1\}$. Such generators are described in [5]–[7], where they are designated $F(r, s, \diamond)$ generators. Methods for establishing periods are in [7].

While examples of generators of each of the three standard methods described above are widely used and —for most purposes—work quite well, it is worth considering new methods. All standard generators (with the exception of lagged-Fibonacci using multiplication) fail one or more stringent tests of randomness such as those described in [6], and most implementations of them have periods too short for the huge samples that current computer speeds make possible.

Note, however, that standard generators can be made to have very long periods: shift-register generators $\beta, \beta T, \beta T^2, \dots$ with β a seed vector of hundreds or thousands of bits; lagged-Fibonacci generators such as $x_n = x_{n-607} - x_{n-243} \bmod 2^{32}$ with period some 2^{607+32} for a set of r 32-bit seed values; extended congruential generators of the form $x_n = a_1 x_{n-1} + a_2 x_{n-2} + \cdots + a_k x_{n-k} \bmod p$ exist for any prime p, with period p^k. But most such extensions have drawbacks—the problem of manipulating β's hundreds of bits long for shift-register generators, or the high cost of arithmetic modulo a prime for extended congruential generators. Lagged-Fibonacci generators using addition, subtraction or multiplication modulo 2^{32} have few such drawbacks, but, to put provide a comparison: the generator $x_n = x_{n-607} - x_{n-243} \bmod 2^{32}$ mentioned above will have period about 2^{607+32}, while an analogous generator of the new type described below will have period about $2^{607\times 32}$.

For these reasons, we offer the class of add-with-carry and subtract-with-borrow generators we now proceed to define.

2. The new class: Add-with-carry generators.

We introduce add-with-carry generators with a simple example. Consider the classical Fibonacci sequence

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \dots,$$

with each element the sum of the previous two. If we take this sequence mod 10, we have an example of a lagged-Fibonacci sequence with lags $r = 2$ and $s = 1$ and binary operation $v \diamond w = v + w \bmod 10$:

$$0, 1, 2, 2, 3, 5, 8, 3, 1, 4, 5, 9, 4, 3, 7, \dots .$$

The information description of the sequence is $x_n = x_{n-2} + x_{n-1} \bmod 10$, but to formally describe it and define and establish its period we need the finite set X of 1×2 vectors $x = (x_1, x_2)$ with elements reduced residues of 10 and the iterating function f defined by $f(x_1, x_2) = (x_2, x_1 + x_2 \bmod m)$. Since f has an inverse, for any initial vector $x \in X$ the sequence

$$x,\ f(x),\ f^2(x),\ f^3(x), \dots$$

is strictly periodic. Depending on the initial vector x, there is a longest cycle of period 60 and shorter cycles of periods 1, 3, 4, 12 and 20. Each period is the least common multiple (lcm) of the periods for moduli 2 and 5.

Now consider the add-with-carry version of this generator. We assign two initial values, say 0, 1, and an initial "carry bit," say 0. Then each new digit is the sum of the previous two digits *plus the carry bit*. The result is taken mod 10 and the next carry bit set to 1 or 0 according to whether or not the sum exceeds 10. Using a superscript to indicate the carry bit, the sequence of digits becomes

$$0, 1^0, 1^0, 2^0, 3^0, 5^0, 8^0, 3^1, 2^1, 6^0, 8^0, 4^1, 3^1, 8^0, 1^1, 0^1, 2^0, \ldots .$$

Formally, as before, we have a sequence of iterates x, $f(x)$, $f^2(x), \ldots$. But now our x's come from the set X of 1×3 vectors $x = (x_1, x_2, c)$ with x_1, x_2 reduced residues of 10 and c the "carry bit," 0 or 1. Then the iterating function f is

$$f(x_1, x_2, c) = \begin{cases} (x_2, x_1 + x_2 + c, 0) & \text{if } x_1 + x_2 + c < 10, \\ (x_2, x_1 + x_2 + c - 10, 1) & \text{if } x_1 + x_2 + c \geq 10. \end{cases}$$

For initial vectors $x = (x_1, x_2, 0)$ with $x_1 < x_2$ or $x = (x_1, x_2, 1)$ with $x_1 > x_2$ the sequence of iterates x, $f(x)$, $f^2(x), \ldots$ is strictly periodic with period 108. If the initial vector x is not of those two types and not $(0, 0, 0)$ or $(9, 9, 9)$, then the sequence beginning with $f(x)$ is strictly periodic with period 108, but the "seed" vector x may not reappear in the sequence. We develop rules for finding the period and assigning seed vectors for add-with-carry generators below.

As with lagged-Fibonacci sequences, a whole class of such generators can be created by altering the lags from the values $r = 2$ and $s = 1$ used in the previous example. The general add-with-carry generator has a base b, lags r and s with $r > s$, a seed vector $x = (x_1, x_2, \ldots, x_r, c)$ with elements "digits" of the base b. Then the generated sequence is x, $f(x)$, $f^2(x)$, $f^3(x), \ldots$ with

$$f(x_1, \ldots, x_r, c) = \begin{cases} (x_2, \ldots, x_r, x_{r+1-s} + x_1 + c, 0) & \text{if } x_{r+1-s} + x_1 + c < b, \\ (x_2, \ldots, x_r, x_{r+1-s} + x_1 + c - b, 1) & \text{if } x_{r+1-s} + x_1 + c \geq b. \end{cases}$$

With appropriately chosen base b, lags r and s and seed vector x, the generated sequence x, $f(x)$, $f^2(x), \ldots$ will be periodic with period $b^r + b^s - 2$, as well be shown in Section 4. These generators have extremely long periods. For example, when b is near 2^{32}, each base-b "digit" is a computer word, and with r around 20 or so, then periods of some 2^{640} are attainable, at the cost of only r memory locations and simple computer arithmetic. (The add-with-carry instruction is basic to all CPU instruction sets.)

2.1. *The complementary add-with-carry generator.* We mention here another kind of add-with-carry generator. It arose as a generator whose period filled a gap in an otherwise complete set of rules for the new generators; see Section 4.4. We call it the *complementary add-with-carry* generator, and its rules for forming the element x_n and associated carry c are

$$\text{if } t = x_{n-r} + x_{n-s} + c < b, \text{ then } x_n = b - 1 - t \text{ and } c = 0$$
$$\text{else } x_n = 2b - 1 - t \text{ and } c = 1.$$

In short: form x_n and c as in the plain add-with-carry generator, but return the complementary digit $b - 1 - x_n$ rather than x_n itself. This rule leads to sequences with periods $b^r + b^s$.

3. The new class: Subtract-with-borrow generators. We illustrate a subtract-with-borrow generator with a numerical example using the same parameters as the add-with-carry generator above (base 10, lags $r = 2$, $s = 1$), but now each new digit is a difference: the lag-1 digit is subtracted from the lag-2 digit *and the carry bit is subtracted as well*. If the result is positive, it becomes the new element with carry bit set to 0; if negative, 10 is added and the new carry bit is set to 1. Thus, with seeds 0, 1 and initial carry 0, the sequence becomes

$$0, 1^0, 9^1, 1^1, 7^0, 4^1, 2^0, 2^0, 0^0, 2^0, 8^1, 3^1, 4^0, 1^1, 2^0, 9^1, 2^1, 6^0, \dots .$$

Formally, for this generator, X is the set of 1×3 vectors $x = (x_1, x_2, c)$, with x_1, x_2 reduced residues for base 10 and $c \in \{0, 1\}$. But now the iterating function f is

$$f(x_1, x_2, c) = \begin{cases} (x_2, x_1 - x_2 - c, 0) & \text{if } x_1 - x_2 - c \geq 0, \\ (x_2, x_1 - x_2 - c + 10, 1) & \text{if } x_1 - x_2 - c < 0. \end{cases}$$

This makes precise the meaning of the generator we informally describe by $x_n = x_{n-2} - x_{n-1} - c$.

Since the order of subtraction matters, a different generator would be formed by $x_n = x_{n-1} - x_{n-2} - c$. More generally, the informal rules for subtract-with-borrow generators with lags r and s are $x_n = x_{n-r} - x_{n-s} - c$ and $x_n = x_{n-s} - x_{n-r} - c$. (Here and throughout, we assume for the two lags r and s that $r > s$.) To formally define the generators and establish their periods we have the finite set X of $1 \times (r + 1)$ vectors $x = (x_1, x_2, \dots, x_r, c)$ with the x's reduced residues of some base b and the iterating function f defined for the $x_n = x_{n-s} - x_{n-r} - c$ case by

$$f(x_1, \dots, x_r, c) = \begin{cases} (x_2, \dots, x_r, x_{r+1-s} - x_1 - c, 0) & \text{if } x_{r+1-s} - x_1 - c \geq 0, \\ (x_2, \dots, x_r, x_{r+1-s} - x_1 - c + b, 1) & \text{if } x_{r+1-s} - x_1 - c < 0 \end{cases}$$

and with f for the $x_n = x_{n-r} - x_{n-s} - c$ case by

$$f(x_1, \dots, x_r, c) = \begin{cases} (x_2, \dots, x_r, x_1 - x_{r+1-s} - c, 0) & \text{if } x_1 - x_{r+1-s} - c \geq 0, \\ (x_2, \dots, x_r, x_1 - x_{r+1-s} - c + b, 1) & \text{if } x_1 - x_{r+1-s} - c < 0. \end{cases}$$

With appropriately chosen base b, lags r and s and seed vector x the generated sequence $x, f(x), f^2(x), \dots$ will be strictly periodic with period $b^r - b^s$ for the $x_n = x_{n-s} - x_{n-r} - c$ generator and period $b^r - b^s - 2$ for $x_n = x_{n-r} - x_{n-s} - c$.

4. Periods of the new generators. The fundamental result for establishing periods comes from recognizing that these generators behave very much like the operation of long addition with carry (that some of us in the pre-calculator age learned in school). Once this addition is explicitly written, it is easy to recognize that the sequence of digits formed by the add-with-carry or subtract-with-borrow operation is, in reverse order, the same as the sequence of digits of the base-b expansion of a proper fraction $k/(b^r \pm b^s \pm 1)$. Here r, s and b are the lags and the base, respectively, and choices of $\pm$ depend on the particular generator.

4.1. *Results from number theory.* We need some background material from number theory to establish periods of the new class of generators. This elementary material has been known for hundreds of years, but it is seldom mentioned in modern books. We summarize it here. It concerns the decimal expansions of fractions—expansions to a base b, rather than the customary base 10—but we illustrate with the more familiar base 10.

Let the modulus m be chosen and consider the group G of $\phi(m)$ reduced residues of m relatively prime to m. For k in G we want the base-b expansion of k/m. That expansion is strictly periodic with period the order of b in the group G. (This requires the assumption that $b \in G$.) The cyclic subgroup generated by b partitions G into cosets. Two elements g and h of G are equivalent (belong to the same coset) if $g = hb^j$ for some j. If two elements g, h belong to the same coset, then g/m and h/m have the same base-b expansion with period the order of b, except that the "digits" in their periods are cyclic permutations of one another.

EXAMPLE. Modulus $m = 39$, base $b = 10$. The powers of 10 mod m generate the cyclic subgroup $\{1, 10, 22, 25, 16, 4\}$, so the order of 10 for modulus 39 is 6. Successive elements k of that subgroup have a common set of digits in the period-6 decimal expansion of k/m, each a cyclic permutation of the previous one:

$$1/39 = 0.025641025\ldots, \quad 10/39 = 0.256410256\ldots, \quad 22/39 = 0.564102564\ldots,$$

$$25/39 = 0.641025641\ldots, \quad 16/39 = 0.410256410\ldots, \quad 4/39 = 0.102564102\ldots.$$

Now choose an element not in the first coset, say 2. Its coset is $\{2, 20, 5, 11, 32, 8\}$, and ratios k/m with k from that coset all have the same digits in their period-6 decimal expansions, shifted by one because of successive multiplications by the base 10:

$$2/39 = 0.051282051\ldots, \quad 20/39 = 0.512820512\ldots, \quad 5/39 = 0.128205128\ldots,$$

$$11/39 = 0.282051282\ldots, \quad 16/39 = 0.820512820\ldots, \quad 8/39 = 0.205128205\ldots.$$

For our purposes, we want to choose primes m for which b is a primitive root. Then the period is $m - 1$ for the base-b expansion of every proper fraction k/m.

4.2. *Periods of the new generators*: *Add-with-carry.* We now prove that *the period of the add-with-carry sequence that produces "digits" of the base b by means of the relation $x_n = x_{n-r} + x_{n-s} + c \bmod b$ is the period of the base-b expansion of k/m for some k in $1 \le k < m$ and $m = b^r + b^s - 1$.*

To fix ideas, consider the case $r = 2$, $s = 1$ for modulus $b = 10$: $x_n = x_{n-2} + x_{n-1} + c \bmod 10$, initialize with the seed digits 1, 2 and initial carry $c = 0$. The sequence is $1, 2, 3, 5, 8, 3, 2, 6, 8, 4, 3, 8, \ldots$. Now the idea behind the proof is that digits of the sequence that are $r - s$ positions apart are added, with carry, to form the next digit, moving left to right. We are all familiar with a similar arithmetic operation when adding two large integers, except that addition-with-carry moves right to left. To illustrate the analogy, let I be the integer whose digits are the first twelve of our sample sequence, *in reverse order*:

$$I = 834862385321.$$

Now shift I left $r - s = 1$ positions (form $10I$) and add it to I, getting

$$\begin{aligned} I &= \mathbf{8348623853}21 \\ bI &= \underline{\mathbf{8348623853}210} \\ I + bI &= 91\mathbf{8348623853}1\,. \end{aligned}$$

Because of the rule for forming the sequence, there will be a substring of digits common to each of the three levels; they are indicated in boldface. Let S be the integer formed by those digits—in this case, $S = 8348623853$. The appearance of S in each of the three levels of the sum enables us to develop a simple linear equation for S:

$$10^2S + 21 + 10^3S + 210 = 91 \times 10^{11} + 10^1S + 1,$$

leading to

$$(10^2 + 10 - 1)S = 91 \times 10^{10} - 23.$$

Thus

$$S = 10^{10}\frac{91}{109} - \frac{23}{109}.$$

Now S is an integer, so the fractional part of $10^{10}(91/109)$ must cancel that of $23/109$, and it follows that S is the integer part of $10^{10}(91/109)$; that is, S's digits are the first 10 digits of the decimal expansion of $91/109$:

$$\frac{91}{10^2 + 10^1 - 1} = 0.8348623853\ldots .$$

We may apply such an argument to the reversed digits in an arbitrarily long finite string formed by $x_n = x_{n-r} + x_{n-s} + c \bmod b$. The period of the sequence will be the period of the base-b expansion of a proper fraction of the form k/m with $m = b^r + b^s - 1$. The integer k will be that formed by the r digits on the left of the string S in the third row of the sum. When k is relatively prime to m the period will be the order of b for modulus m. Making m a prime ensures this, of course.

Here are some other examples. In the first two the sequence is

$$x_n = x_{n-4} + x_{n-2} + c \bmod 10, \qquad \text{seed } 7493 \to 74936852218305888372\ldots$$

with initial carry $c = 0$. Take, say, the first 20 digits of the sequence and write them in reverse order to get the integer I. Then form $b^{r-s}I$ by shifting I left two positions, bringing in two zeros. Then add. Because of the rule for forming new digits, the three levels will have a common string (indicated in boldface):

$$\begin{aligned} I &= \mathbf{2738885038122586}3947 \\ b^2I &= \underline{\mathbf{2738885038122586}394700} \\ I + b^2I &= 2766\mathbf{2738885038122586}47\,. \end{aligned}$$

Let S be the integer formed by those digits. The presence of S in each of the three levels provides a linear equation,

$$b^4S + 3947 + b^6S + 394700 = 2766b^{18} + b^2S + 47,$$

leading to

$$(b^4 + b^2 - 1)S = 2766b^{16} - 3908,$$

and thus

$$S = \frac{2766b^{16}}{b^4 + b^2 - 1} - \frac{3908}{b^4 + b^2 - 1}.$$

Since S is an integer, the fractional part of the first term must cancel the second, and thus S is the first 16 digits of the expansion of $2766/(b^4 + b^2 - 1)$:

$$\frac{2766}{10^4 + 10^2 - 1} = 0.2738885038122586\dots\,.$$

Suppose instead of 20, we take 15 digits of the above sequence and form an integer I by writing them in reverse order. We then add $b^{r-s}I$ to get this tableaux with the digits of S in boldface:

$$\begin{aligned} I &= \mathbf{85038122586}3947 \\ b^2I &= \underline{\mathbf{85038122586}394700} \\ I + b^2I &= 8588\mathbf{85038122586}47\,. \end{aligned}$$

The linear equation in S that results from this sum is

$$b^4S + 3947 + b^6S + 394700 = 8588b^{13} + b^2S + 47,$$

from which, because S is an integer, we conclude that $S = 8588b^{11}/(b^4 + b^2 - 1)$; that is, the digits of S are the first 11 digits in the expansion of 8588/10099:

$$\frac{8588}{10^4 + 10^2 - 1} = 0.85038122586\dots\,.$$

Finally, here is one more tableaux with base $b = 6$ and lags $r = 6$ and $s = 3$:

$$x_n = x_{n-6} + x_{n-3} + c \bmod 6, \qquad \text{seed } 153024 \to 153024112230442\dots,$$

$$\begin{aligned} I &= \mathbf{244032211}420351 \\ b^3I &= \underline{\mathbf{244032211}420351000} \\ I + b^3I &= 244320\mathbf{244032211}351\,, \end{aligned}$$

$$\frac{244320}{6^4 + 6^3 - 1} = 0.244032211\dots \text{ base } 6.$$

4.3. *Periods of the generators*: *Subtract-with-borrow*. For sequences generated by $x_n = x_{n-r} - x_{n-s} - c \bmod b$, the period is also the period of a proper fraction k/m, where this time $m = b^r - b^s - 1$. This may be established in a manner similar to that for add-with-carry sequences: Choose r starting values and generate a sequence of arbitrary length. As before, let I be the integer formed by putting those digits in reverse order. For example, with $b = 10$, $r = 5$, $s = 3$ and starting values 5, 9, 7, 7, 7 the reversed string of 15 digits yields $I = 304285901877795$. Shift I left by s positions (multiply by b^s) and add to get

$$\begin{aligned} I &= 30428\mathbf{59018777}95 \\ b^sI &= \underline{30428\mathbf{59018777}95000} \\ I + b^sI &= 304\mathbf{59018777}9672795\,. \end{aligned}$$

The boldface string $S = 59018779$ appears in each line. Cancel the leading 304 and the trailing 5 to get a simplified equation for S:

$$30428b^9 + S + 28b^{12} + b^3S + 500 = b^5S + 67279.$$

Thus

$$58428b^9 - 64779 = (b^5 - b^3 - 1)S$$

and, with $m = b^5 - b^3 - 1 = 98999$,

$$S = 10^9\frac{58428}{98999} - \frac{66779}{98999}.$$

Since S is an integer, it must be the integer part of the first term; that is, the digits of S are the leading nine digits of the decimal expansion of $58428/98999$: $S = 590187779$.

We may apply such an argument to the reversed string of an arbitrary length sequence with r initial values and, for $n > r$, $x_n = x_{n-r} - x_{n-s} - c \bmod b$. Its period will be the period of the base-b expansion of a proper fraction k/m, with $m = b^r - b^s - 1$. (The particular value of k changes with the length of the string used to form I. It is of no particular importance—the actual value is that of the trailing 5 digits of $J + b^sJ$, where J is the integer formed by the leading r digits of I. The important point is that the linear equation always has S with coefficient b^dm, for some d and $m = b^r - b^s - 1$, so that the solution for the integer S is the first d digits of a proper fraction k/m for some k.)

For the case $x_n = x_{n-s} - x_{n-r} - c$ with $r > s$ a similar derivation holds, except that I is added to b^rI. The common string S has a solution of the form $S = b^d(k/m) + \theta$ with $m = b^r - b^s + 1$ and $0 < \theta < 1$. Then the base-b digits of S are again obtained from the leading digits of the base-b expansion of a proper fraction k/m. We illustrate with a final tableaux:

$$x_n = x_{n-3} - x_{n-5} - c \bmod 10,$$

$$\text{seed } 26479 \to 26479215512466242679 1534\ldots,$$

$$I = 976\mathbf{242664215512974}62$$

$$b^rI = \underline{976\mathbf{242664215512974}6200000}$$

$$I + b^rI = 97625\mathbf{242664215512974}97462\,,$$

$$(b^5 - b^3 + 1)S = 24024b^{15} - 6102538,$$

$$\frac{24024}{10^5 - 10^2 + 1} = 0.242664215512974\ldots\,.$$

The rule for forming k may be deduced by noting that $24024 = 25000 - 976$.

4.4. *Periods*: *The complementary add-with-carry generator*. We have three kinds of generators whose periods are those of the base-b expansions of proper fractions k/m, with $m = b^r + b^s - 1$ for $x_n = x_{n-r} + x_{n-s} + c \bmod b$; $m = b^r - b^s - 1$ for $x_n = x_{n-r} - x_{n-s} - c \bmod b$; and $m = b^r - b^s + 1$ for $x_n = x_{n-s} - x_{n-r} - c \bmod b$. Thus all possible forms $m = b^r \pm b^s \pm 1$ occur, except $m = b^r + b^s + 1$. With faith in the beauty and completeness of mathematics, we thought there should be a generator for that case as well, and it turned out that there is—the complementary add-with-carry generator described in 2.1.

This generator provides sequences with periods that of the base-b expansion of proper fractions of the form k/m with $m = b^r + b^s + 1$. Readers seeking to verify this may construct tableaux as above: Form I as the digits of a sequence in reverse order, add b^rI to get a third row. Then the *complement* of the third row (replace a digit d by $b - 1 - d$) will have a string S in common with the first two rows. This will provide a linear equation in S with coefficient $b^d(b^r + b^s + 1)$, and thus the integer S will be the first d digits of the base-b expansion of some $k/(b^r + b^s + 1)$.

Because of its slightly more complicated generating procedure, the complimentary add-with-carry generator does not seem as desirable as the other three. However, it does have the redeeming feature that primes of the form $m = b^r + b^s + 1$ make factoring $m - 1$ easier. Such factoring is essential in establishing that b is a primitive root.

4.5. *Summary for periods and periodic seed vectors*. The periods established above for the four kinds of generators are always those of the base-b expansion of a proper fraction k/m. Indeed, the base-b "digits" generated are those of the expansion of k/m in reverse order. Here $m = b^r \pm b^s \pm 1$. When implementing these generators, we will always choose r, s so that m is a prime. Then the period of the base-b expansion of k/m is the order of b in the group of residues of m, and that order is $m - 1$ if b is a primitive root. So we seek primes of the form $b^r \pm b^s \pm 1$ with b a primitive root. Occasionally, for particularly desirable b's such as $b = 2, 2^8, 2^{16}, 2^{32}$ we will have to settle for primes m for which the order of b is nearly, but not quite, $m - 1$.

Any starting (seed) vector of r digits and initial carry c will produce a sequence that is ultimately periodic, but it may not be strictly periodic: After a few iterations (at most r), the periodic cycle begins. We say that a seed vector is "periodic" if it produces a strictly periodic sequence: The first vector to be repeated in the sequence x, $f(x)$, $f^2(x), \ldots$ is the seed vector x itself. Finding conditions which ensure that a seed vector is periodic is pretty much an academic exercise, done out of curiosity. We emphasize that *whatever the seed vector, except for the two trivial seeds, the add-with-carry and subtract-with-borrow sequences become periodic after a few iterations of the generating function, and the periods are the order of the base b for the appropriate modulus* $m = b^r \pm b^s \pm 1$.

The question of whether a seed vector produces a strictly periodic sequence or one that becomes periodic after a few "stabilizing" iterations, is of no significance for practical applications of the new generators. But the puzzle of exactly which seed vectors produce strictly period sequences is an interesting challenge that we undertook to solve. The following listing summarizes rules we found for periodic seed vectors and periods for subtract-with-borrow and add-with carry generators, with base b, lags r and s, $r > s$. If r and s are chosen so that m is prime and b is a primitive root of m, the (long) period of the sequence will be $m - 1$. [There are two short periods, each of length 1, for the trivial seed vectors $(0, \ldots, 0, 0)$ and $(b - 1, \ldots, b - 1, 1)$.] Periodic seed vectors have the form $(x_1, x_2, \ldots, x_r, c)$ with rules for their formation given for each method. When we write a succession of symbols such as $x_8 x_7 \cdots x_1$ we mean the integer for which that is the base-b representation.

METHOD 1. $x_n = x_{n-s} - x_{n-r} - c \bmod b$; $m = b^r - b^s + 1$.

$$c = 0 \text{ and } x_r \cdots x_{s+1} < x_{r-s} \cdots x_1,$$
$$c = 1 \text{ and } x_r \cdots x_{s+1} > x_{r-s} \cdots x_1.$$

METHOD 2. $x_n = x_{n-r} - x_{n-s} - c \bmod b$; $m = b^r - b^s - 1$.

$$c = 0 \text{ and } x_r \cdots x_{s+1} + x_{r-s} \cdots x_1 < b^{r-s} - 1,$$
$$c = 1 \text{ and } x_r \cdots x_{s+1} + x_{r-s} \cdots x_1 > b^{r-s} - 1.$$

METHOD 3. $x_n = x_{n-r} + x_{n-s} + c \bmod b$; $m = b^r + b^s - 1$.

$$c = 0 \text{ and } x_r \cdots x_{s+1} \geq x_{r-s} \cdots x_1,$$
$$c = 1 \text{ and } x_r \cdots x_{s+1} \leq x_{r-s} \cdots x_1.$$

METHOD 4. Form x_n and carry as in Method 3, but replace x_n by its complement, $b - 1 - x_n$; $m = b^r + b^s + 1$.

$$c = 0 \text{ and } x_r \cdots x_{s+1} + x_{r-s} \cdots x_1 \leq b^{r-s} - 1,$$
$$c = 1 \text{ and } x_r \cdots x_{s+1} + x_{r-s} \cdots x_1 \geq b^{r-s} - 1.$$

We conclude this section with examples of rules for periodic starting vectors $(x_1, x_2, x_3, x_4, x_5, c)$ for the particular case $b = 10$ and $r = 5$, $s = 2$. In order that the sequence be strictly periodic, the seed vector must have one of two possible forms:

1. For the generator $x_n = x_{n-s} - x_{n-r} - c \mod 10$,

$$(x_1, x_2, x_3, x_4, x_5, 0) \text{ with integers } x_5x_4x_3 > x_3x_2x_1,$$

$$(x_1, x_2, x_3, x_4, x_5, 1) \text{ with integers } x_5x_4x_3 < x_3x_2x_1.$$

2. For the generator $x_n = x_{n-r} - x_{n-s} - c \mod 10$,

$$(x_1, x_2, x_3, x_4, x_5, 0) \text{ with integers } x_5x_4x_3 + x_3x_2x_1 < 999,$$

$$(x_1, x_2, x_3, x_4, x_5, 1) \text{ with integers } x_5x_4x_3 + x_3x_2x_1 > 999.$$

(a) For the generator $x_n = x_{n-r} + x_{n-s} + c \mod 10$,

$$(x_1, x_2, x_3, x_4, x_5, 0) \text{ with integers } x_5x_4x_3 \le x_3x_2x_1,$$

$$(x_1, x_2, x_3, x_4, x_5, 1) \text{ with integers } x_5x_4x_3 \ge x_3x_2x_1.$$

(b) For the generator $x_n = 9 - x_{n-r} - x_{n-s} - c \mod 10$,

$$(x_1, x_2, x_3, x_4, x_5, 0) \text{ with integers } x_5x_4x_3 + x_3x_2x_1 \le 999,$$

$$(x_1, x_2, x_3, x_4, x_5, 1) \text{ with integers } x_5x_4x_3 + x_3x_2x_1 \ge 999.$$

If the sequence starts with any other seed vector, the sequence will not be strictly periodic, but a *rho* sequence: an initial string of a few elements followed by the cyclic part. The cycles will always be those of the base-b expansion of some proper fraction k/m, in reverse order.

5. Structure of the sequences. In this section we clarify what we meant by "provable uniformity for full sequences," mentioned in the abstract. We begin with comments on seed values and periods. An ideal generator should have period as great as the number of possible choices for seed values. Then, if the seed values are $x_1, x_2, \dots, x_r$ and the sequence is strictly periodic, every possible r-tuple of x's will appear in the full sequence—a desirable uniformity property. Except for trivial cases of little interest, the lagged-Fibonacci generators—the current record holders for long periods—do not have this property. The lagged-Fibonacci generators $F(r, s, - \mod 2^{32})$, $F(r, s, * \mod 2^{32})$ or $F(r, s, - \mod 1)$ have periods on the order of $2^{32+r}, 2^{30+r}, 2^{24+r}$, far short of the ideals of 2^{32r}, 2^{30r} or 2^{24r} that are the number of possible choices of seed values. [Nonetheless, their periods are still far longer than those for $F(r, s, \oplus)$ generators using exclusive-or, for which the period is at most 2^r, whatever the word size.]

The new generators developed here require r seed digits from a base b, and their periods differ by an insignificant fraction (b^s/b^r) from b^r, the set of all possible r-tuples. Thus, for example, the subtract-with-borrow generator developed in the next section, $x_n = x_{n-22} - x_{n-43} - c \mod 2^{32} - 5$, has base $b = 2^{32} - 5$ and period $b^{43} - b^{22}$. It requires 43 seed values, each a 32-bit integer in the range 0 to $b - 1$. Every possible 43-tuple of base-b digits will appear in a full period, except for the missing fraction $1/b^{21} \approx 10^{-202}$ of the possible 43-tuples.

Because any set of 43 seed values can be used to initiate the sequence, it follows that every 42-tuple of digits will appear b times in the full period, except for those associated with the insignificant fraction of missing 43-tuples, and so on: All k-tuples for $k = 43, 42, \ldots, 2, 1$ will appear in a full period with frequencies consistent (indeed, *too* consistent) with uniformity. But, of course, it is impossible to use any but a small portion of such immense periods. Presumably, the frequencies will have local departures from uniformity that are consistent with randomness; only tests can verify that.

Rather than dealing with such immense periods, it is sometimes helpful to examine all of the elements from one of the new generators with a more modest period. Consider the prime $10^5 - 10^2 + 1 = 99901$, for which 10 is a primitive root. The subtract-with-borrow sequence $x_n = x_{n-2} - x_{n-5} - c \bmod 10$ has period 99900. It will be strictly periodic if, and only if, the seed vector is one of the two types:

$$(x_1, x_2, x_3, x_4, x_5, 0) \text{ with integers } x_5x_4x_3 > x_3x_2x_1,$$

$$(x_1, x_2, x_3, x_4, x_5, 1) \text{ with integers } x_5x_4x_3 < x_3x_2x_1.$$

There are $10^5 = 100{,}000$ possible 5-tuples $x_1x_2x_3x_4x_5$, but only 99,900 of them appear in the full-period sequence. Which ones are missing? Evidently the 5-tuples which cannot be used to form a periodic seed vector; that is, 5-tuples with $x_5x_4x_3 = x_3x_2x_1$. These have the form *xyxyx*, 100 in number. Other than those 100 exceptions, every 5-tuple appears exactly once in the full period of 99,900 5-tuples. Thus the sequence has nearly full-period uniformity, in that virtually all of the possible 5-tuples appear exactly once, and only a few —those of the form *xyxyx*—do not appear. Furthermore, except for those that could arise from 5-tuples of that form, every 4-tuple appears the right number of times in the full cycle, and so on for 3-tuples, 2-tuples and individual digits.

6. Choosing the base *b* and lags *r* and *s*.

We begin this section with a specific choice for one of the new generators. It is one of the best we have found. It has already been implemented in a widely used generator called RANMAR advocated by James [1] and in a generator specifically tailored for PC's [9].

Discussion of the method for finding suitable parameters and proving that b is a primitive root will illustrate what must be done to find good, practical subtract-with-borrow or add-with-carry generators.

For this generator we have $b = 2^{32} - 5$. The "digits" for base b are then integers that account for virtually all 32-bit computer words. (Even better would be $b = 2^{32}$, so that the digits are exactly the set of 32-bit integers, but for reasons developed below, there are no primes of the required form for which 2^{32} is a primitive root.) The choice $b = 2^{32} - 5$ arose from an extensive search for parameters b, r, s meeting these conditions: b near 2^{32}, r and s

such that $m = b^r - b^s + 1$ is a prime with b a primitive root, r not too small and s not too close to r.

Such searches lead to formidable computing problems. For given b near 2^{32} and with r some 20 or more, one must first find primes of the form $m = b^r - b^s + 1$ and then see if b is a primitive root. Such m's might be in the range 2^{600} to 2^{1800}. Testing for primality is feasible, using Monte Carlo tests, but establishing whether b is a primitive root is much more difficult. That requires factoring $m - 1$. However, hundreds of hours of computing produced this promising subtract-with-borrow generator:

$$x_n = x_{n-22} - x_{n-43} - c \bmod b, \qquad b = 2^{32} - 5 = 4294967291.$$

Given 43 seed values, each a 32-bit integer less than $2^{32} - 5$, the generated sequence will have period $m - 1 = b^{43} - b^{22} \approx 2^{1376}$. To establish this immense period we must prove that $m = b^{43} - b^{22} + 1$ is prime and $b = 2^{32} - 5$ is a primitive root of m. So we factor $m - 1$, which has 17 prime factors: Let g be that factor set:

$$\begin{aligned} g = \{&b, 2, 5, 19, 43, 421, 883, 7057, 9829, 46681, 3650221, 22605091,\\ &447526613551, 1152964457, 7192358279,\\ &1760368045354314379560378839291601799З, p'\}. \end{aligned}$$

Here p' is a prime of 99 digits requiring two lines for display:

$$\begin{aligned} p' = {}&36964737049079490962774762893967802680434669317822\\ &4186677750022102445220940696162576134304437648517. \end{aligned}$$

Since $b^{m-1} = 1 \bmod m$ and $b^{(m-1)/p} \neq 1 \bmod m$ for each prime p that divides $m - 1$, it follows that m is prime and b is a primitive root.

6.1. *Bases a power of* 2. For computer implementation of the new generators, the most desirable bases are powers of 2. Choosing $b = 2^{32}$ makes each digit of the base b a full computer word, a 32-bit integer. Choosing $b = 2$ allows generation of a stream of bits in a single-bit processor or bits 16 or 32 at a time by means of the integer addition or subtraction available in 16- or 32-bit processors with built-in bit-to-bit carry. (It was in response to a request for a random number generator for the Connection Machine, with its 65536 one-bit processors that the methods of this article were developed.)

Unfortunately, there is no prime $m = 2^r \pm 2^s \pm 1$ for which 2 is a primitive root, unless the shorter lag s is 1 or 2. The reason for this is that 2 is always a quadratic residue of primes $m = 2^r \pm 2^s \pm 1$ if $s > 2$, since $m = \pm 1 \bmod 8$, and of course a quadratic residue cannot be a primitive root.

Furthermore, if 2 is a quadratic residue of m, then 2 has a square root mod m and thus so have $2^{24}, 2^{32}$ and all the powers of 2. So we must give up hope of full-period add-with-carry or subtract-with-borrow generators for bases $2, 2^{16}, 2^{24}, 2^{32}$ and so on unless we are willing to let the shorter lag s be 1 or 2. However, there are good choices for r and s for those bases where the period is almost the maximum possible. We give a few of them now for base $b = 2^{24}$.

6.2. *Base* $b = 2^{24}$. This base is particularly attractive for use with the subtract-with-borrow generators because it makes possible the generation of floating-point numbers directly, without the usual method of generating an integer and then dividing by the modulus. This feature was exploited in the "universal" generator described in [9]. The choice $b = 2^{24}$ allows direct generation of computer reals with 24-bit fractions—the most frequent size for single precision.

Since $b = 2^{24}$ cannot be a primitive root of any $m = b^r \pm b^s \pm 1$ except when $s = 1$ or 2, we can have no useful generators with full period $m - 1$. But if m is prime and the order of b is $(m-1)/j$ for small j, then we may still have a useful generator with immensely long periods. Here are three examples for $b = 2^{24}$: $m = b^{24} - b^{10} + 1$, $m = b^{25} - b^{11} + 1$, $m = b^{39} - b^{25} + 1$. For these primes m we are able to factor $m - 1$ and show that the order of b is not significantly smaller than $m - 1$. Here is how to do it: For each of these primes m, the prime factors of $m - 1$ are

$$2, 3, 5, 7, 13, 17, 29, 43, 97, 113, 127, 241, 257, 337, 673, 1429, 2017, 3361,$$
$$5153, 5419, 14449, 15790321, 25629623713, 88959882481,$$
$$54410972897, 1538595959564161.$$

[Since, in each case, $m - 1 = b^k(b^7 - 1)(b^7 + 1)$, we "only" need the prime factors of $2^{168} - 1$ and $2^{168} + 1$, listed, with 2, above.] With these prime factors of $m - 1$ we may find the order of $b = 2^{24}$ for modulus m. In none of these cases does b have order $m - 1$, but the order is large enough to provide sequences with extremely long periods. These three generators are among those recommended in Table 2 in the summary, Section 9. They are:

1. $r = 24$, $s = 10$, $b = 2^{24}$, $m = b^{24} - b^{10} + 1$. The order of b for modulus m is $(m-1)/48$. For any k in $1 \le k < m$ the base-b expansion of k/m has period $(m-1)/48$, with "digits" in the period a cyclic permutation of one of 48 possible sets of $(m-1)/48$ digits—in this case, about $2^{572}/3$.
2. $r = 25$, $s = 11$, $b = 2^{24}$, $m = b^{25} - b^{11} + 1$. The order of b for modulus m is $(m-1)/336$. For any k in $1 \le k < m$ the base-b expansion of k/m has period $(m-1)/336$, with "digits" in the period a cyclic permutation of one of 336 possible sets of $(m-1)/336$ digits—in this case, about $2^{596}/21$.
3. $r = 39$, $s = 25$, $b = 2^{24}$, $m = b^{39} - b^{25} + 1$. The order of b for modulus m is $(m-1)/672$. For any k in $1 \le k < m$ the base-b expansion of k/m has period $(m-1)/672$, with "digits" in the period a cyclic permutation of one of 672 possible sets of $(m-1)/672$ digits— in this case, about $2^{931}/21$.

In the above cases, each "digit" is a 24-bit integer or the 24-bit fraction of a floating point number, and only 24 or 25 or 39 seed values are required to initiate sequences of such lengths.

6.3. *Choices* r, s *when* $b = 2$. We conclude this section with some potentially useful pairs r, s for which $m = 2^r - 2^s + 1$ is prime. It is, in general, very difficult to find the order of 2 mod m when m is on the order of 2^{600} or more, since it requires factoring $m - 1$. But if we take advantage of primes of

TABLE 1
Pairs r and s for which $m = 2^r - 2^s + 1$ is prime

3, 1	23, 20	57, 38	123, 120	193, 190	323, 320	467, 436
4, 2	25, 12	59, 28	125, 18	201, 182	325, 198	477, 446
5, 2	25, 18	61, 54	145, 140	203, 201	329, 222	481, 374
7, 4	29, 26	65, 60	149, 42	211, 209	349, 242	487, 468
7, 5	31, 24	68, 66	153, 122	217, 110	355, 353	511, 504
8, 6	32, 30	69, 8	161, 72	221, 214	361, 300	536, 534
9, 2	33, 20	73, 68	161, 142	221, 216	393, 390	537, 476
9, 6	37, 18	79, 72	165, 38	251, 232	410, 408	553, 492
10, 8	38, 36	95, 92	169, 138	253, 192	421, 416	565, 558
13, 8	41, 10	101, 70	177, 174	263, 232	425, 412	589, 558
14, 12	43, 41	103, 84	183, 180	269, 208	435, 432	629, 522
17, 14	49, 30	105, 74	185, 58	271, 144	440, 438	785, 178
19, 12	53, 34	109, 78	187, 180	278, 276	441, 380	847, 240
20, 18	53, 50	117, 56	191, 64	293, 290	451, 324	1751, 472
23, 4	55, 52	121, 114	191, 189	309, 248	457, 368	

the form $2^p - 1$ (Mersenne primes) we narrow our search to primes of the form $m = 2^r - 2^s + 1$ with $r = p + s$. For in such cases $m - 1 = 2^s(2^p - 1)$ so that factoring $m - 1$ and determining the order of 2 mod m is easy. The results of the search are in Table 1. A few r, s with s close to r are included in the table for completeness; they are not likely to produce satisfactory random sequences.

7. A practical example for classroom or textbook use. Computer implementations of add-with-carry or subtract-with-borrow generators have immensely long periods and the "digits" for the base b are usually computer words. For expository purposes, it is desirable to have examples with more manageable periods, using more familiar bases b. The example $b = 10$ and $x_n = x_{n-2} - x_{n-5} - c \bmod 10$ was used above. Its period is $10^5 - 10^2$, short enough for examining the full period, but not long enough to apply many tests of randomness. We have found the following add-with-carry generator useful for demonstrations. It uses base $b = 6$ and thus simulates throws of a die:

$$x_n = x_{n-21} + x_{n-2} + c \bmod 6$$

Because $6^{21} + 6^2 - 1$ is a prime for which 6 is a primitive root, this generator will have period $6^{21} + 6^2 - 2 = 21{,}936{,}950{,}640{,}377{,}890$ for any set of 21 seed digits and initial carry c, except for 21 0's and $c = 0$ or 21 5's and $c = 1$.

So, given an initial set of 21 digits for base 6 and an initial carry c, this generator, using very simple arithmetic, will produce a string of some 21×10^{15} "throws" of a die. Every possible set of 21 successive throws will appear in the sequence, with frequencies for shorter strings consistent with uniformity for the full period. To see if the sequence behaves locally as a series of independent throws, one must apply tests.

In addition to all of the tests of randomness mentioned in the next section, which it passed beautifully, we applied a test related to the use of dice for gambling—the *craps test*. In the craps test one plays, say, one million games of craps, using pairs of digits to provide the total on a pair of dice. Counted are: the number of wins (probability 244/495), the number of throws needed to complete the game, the number of "passes" (successive wins) and the frequencies of individual and paired totals on the dice.

The results were indistinguishable from what would be expected from a million games of craps with true dice, consistent with the probabilities for the various aspects of the game—number of wins, duration of the game, frequency of consecutive "passes" and values on the dice. The die was "thrown" 6,747,174 times. There are plenty more throws available; however, the period is $6^{21} + 6^2 - 1 = 21{,}936{,}950{,}640{,}377{,}890$ for the $x_n = x_{n-22} + x_{n-2} + c \bmod 6$ generator, so it could easily run Las Vegas for 10,000 years with no one (but us) the wiser.

8. Tests of randomness. Easily understood examples such as that above for dice serve to illustrate how a simple deterministic process can produce numbers that for many purposes can be taken as random—indeed, for *most* purposes, for it is often very difficult to find applications for which a good random number generator fails to produce results consistent with the underlying probability theory. Any number of tests of randomness may be applied. In the early 1960's, MacLaren and Marsaglia [3] proposed a number of tests for random number generators, and these were taken up and added to by Knuth [2] to become a sort of standard set of tests.

Most generators pass these tests, but several of the standard generators were found to give bad results on more stringent tests such as those described in [6] and [9]. Our research group has a battery of tests called DIEHARD that includes all the standard and the additional tests mentioned in [6] and [9]. The add-with-carry and subtract-with-borrow generators developed here have so far passed all these tests. We will not report specific test results here, but, based on the results we have observed, we think these new methods produce "randomness' at least as well as any other generators. With their simple generating procedures, modest memory requirements and immense periods, they merit consideration for serious Monte Carlo work.

9. Summary and recommendations. We have described new kinds of generators: add-with-carry and subtract-with-borrow. They produce exceptionally long sequences of "digits" $x_1, x_2, \dots$ of a base b. Informally, the generators are of four types:

1. $x_n = x_{n-s} - x_{n-r} - c \bmod b,\ m = b^r - b^s + 1.$
2. $x_n = x_{n-r} - x_{n-s} - c \bmod b,\ m = b^r - b^s - 1.$
3. $x_n = x_{n-r} + x_{n-s} + c \bmod b,\ m = b^r + b^s - 1.$
4. $x_n = b - 1 - x_{n-r} - x_{n-s} - c \bmod b,\ m = b^r + b^s + 1.$

If $r > s$ are chosen so that the designated value m is prime, then the sequence

TABLE 2
Some recommended subtract-with-borrow generators $x_n = x_{n-s} - x_{n-r} - c \bmod b$

Number of seeds and type	Base b	Lags r	s	Number of cycles*	Period of each cycle
847 bits	2	847	240	2	$2^{846} - 2^{239} \approx 10^{255}$
1751 bits	2	1751	472	2	$2^{1750} - 2^{471} \approx 10^{527}$
43 32-bit integers[†]	2^{32}-5	43	22	1	$b^{43} - b^{22} \approx 10^{414}$
37 32-bit integers	2^{32}	37	24	64	$2^{1178} - 2^{762} \approx 10^{354}$
24 32-bit integers	2^{32}	24	19	1536	$\frac{1}{3}(2^{759} - 2^{599}) \approx 10^{228}$
21 32-bit integers	2^{32}	21	6	192	$\frac{1}{3}(2^{666} - 2^{186}) \approx 10^{200}$
48 31-bit integers	2^{31}	48	8	210	$\frac{1}{105}(2^{1487} - 2^{247}) \approx 10^{445}$
39 reals[‡]	2^{24}	39	25	672	$\frac{1}{21}(2^{931} - 2^{595}) \approx 10^{279}$
28 reals[‡]	2^{24}	28	8	144	$\frac{1}{9}(2^{668} - 2^{188}) \approx 10^{200}$
25 reals[‡]	2^{24}	25	11	336	$\frac{1}{21}(2^{596} - 2^{260}) \approx 10^{178}$
24 reals[‡]	2^{24}	24	10	48	$\frac{1}{3}(2^{572} - 2^{236}) \approx 10^{171}$

*Not counting the two trivial cycles of period 1.
[†]Not exceeding $2^{32} - 6$.
[‡]Reals with 24-bit fractions. The floating-point version of the mod 2^{24} operation takes three inputs x, y and c and then produces a new value z and new carry c: $(z, c) = \{$if $x - y - c \geq 0$ then $(x - y - c, 0)$ else $(x - y - c + 1, 2^{-24})\}$.

of x's are the digits, in reverse order, of the base-b expansion of k/m for some $1 \leq k < m$. Formally, the sequence is x, $f(x)$, $f^2(x), \ldots,$ where x is an initial vector of r seed digits and an initial "carry" bit c. Rules for f's that transform x into a new sequence of r digits and an associated carry bit are in Sections 2 and 3, with examples.

Various choices of b, r and s provide for long sequences of bits, bytes, 16-, 31- or 32-bit integers or floating point numbers with 24-, 48- or 64-bit fractions. The extremely long periods of the generators, their small memory requirements and the simple computer operations required, taken with excellent performance on stringent tests of randomness, make subtract-with-borrow and add-with-carry generators worth considering for general Monte Carlo use.

We conclude with Table 2, which gives good choices of r and s for various bases b. Only subtract-with-borrow generators are listed. Add-with-carry generators of equally long periods surely exist, but the periods are so long that only for primes of the form $m = b^r - b^s + 1$ are we able to factor $m - 1$, a necessary step in establishing the period.

REFERENCES

[1] JAMES, F. (1990). A review of pseudorandom number generators. *Comput. Phys. Commun.* **60** 329–344.

[2] KNUTH, D. E. (1981). *The Art of Computer Programming: Volume 2: Seminumerical Algorithms*, 2nd ed. Addison-Wesley, Reading, Mass.

[3] MacLaren, M. D. and Marsaglia, G. (1965). Uniform random number generators. *J. Assoc. Comput. Mach.* **12** 83–89.
[4] Marsaglia, G. (1972). The structure of linear congruential sequences. In *Applications of Number Theory to Numerical Analysis* (Z. K. Zaremba, ed.). Academic, New York.
[5] Marsaglia, G. (1983). Random number generation. In *Encyclopedia of Computer Science and Engineering* (A. Ralston, ed.) 1260–1264. Van Nostrand Reinhold, New York.
[6] Marsaglia, G. (1985). A current view of random number generators, keynote address. In *Proceedings, Computer Science and Statistics: 16th Symposium on the Interface*. North-Holland, Amsterdam.
[7] Marsaglia, G. and Tsay, L. H. (1985). Matrices and the structure of random number sequences. *Linear Algebra Appl.* **67** 147–156.
[8] Marsaglia, G., Narasimhan, B. and Zaman, A. (1990). A random number generator for PC's. *Comput. Phys. Commun.* **60** 345–349.
[9] Marsaglia, G., Zaman, A. and Tsang, W. W. (1990). Toward a universal random number generator. *Statist. Probab. Lett.* **9** 35–39.

Supercomputer Computations Research Institute
and Department of Statistics
The Florida State University
Tallahassee, Florida 32306