Uniform Random Number Generation With Jumping Facilities

C

E. Jack Chen
BASF Corporation, USA

INTRODUCTION

As computer capacities and simulation technologies advance, simulation has become the method of choice for modeling and analysis. The fundamental advantage of simulation is that it can tolerate far less restrictive modeling assumptions, leading to an underlying model that is more reflective of reality and thus more valid, leading to better decisions (Lucas et al. 2015). Simulation studies are typically proceed by transforming in a more or less complicated way of a sequence of numbers between 0 and 1 produced by a pseudorandom generator into an observations of the measure of interest. A facility for generating sequences of pseudorandom numbers is a fundamental part of computer simulation systems. Furthermore, random number generators also play an important role in cryptography. A collection of random variables $x_1, x_2, \ldots, x_n$ is a random sample if they are independent and identically distributed. True random numbers cannot be produced by a deterministic algorithm, and hence, random numbers generated by using a recursive equation are referred to as pseudorandom numbers. The deterministic nature of these techniques is important because it can be reproduced in computations. A facility for generating sequences of pseudorandom numbers is a fundamental part of computer simulation systems. Usually, in practice, such a facility produces a deterministic sequence of values, but externally these values should appear to be drawn independently from a uniform distribution between 0 and 1, i.e., they are independent and statistically indistinguishable from a truly random sequence. Furthermore, multiple independent streams of random numbers are often required in simulation studies, for instance, to facilitate synchronization for variance-reduction purposes, and for making independent replications.

A random number generator (RNG) is an algorithm that starting from an initial seed (or seeds), produces a stream of numbers that behaves as if it were a random sample when analyzed using statistical tests. The RNG is closely related to the Deterministic Random Bit Generators (DRBGs). See L'Ecuyer (1990, 2013) and references therein for more information on RNGs. We describe a portable set of software utilities for uniform random-number generation. It provides for multiple generators (streams) running simultaneously, and each generator (stream) has its sequence of numbers partitioned into many long disjoint contiguous substreams, see L'Ecuyer et al. (2002). Simple procedure calls allow the user to make any generator "jump" ahead/back v steps (random numbers). Implementation issues are discussed. The basic underlying generator CMRG (Combined Multiple Recursive Generator) combines two multiple recursive random number generators with a period length of approximately 2^{191} ($\approx 3.1\times10^{57}$), good speed, and excellent theoretical properties, e.g., the lattice structure, see Kroese et al. (2011) for a list of desired properties.

BACKGROUND

There are a number of methods for generating the random numbers, of which the most popular are the congruential methods (mixed, multiplicative,

DOI: 10.4018/978-1-5225-2255-3.ch111

and additive). The (mixed) linear congruential generators (LCGs) are defined by

$$x_i = \left(ax_{i-1} + c\right) MOD\ m, \quad u_i = \frac{x_i}{m},\ x_0 \in \{1,\ldots,m-1\}\ i > 0.$$

where m (the modulus) is a positive integer (usually a very large primary number), a (the multiplier) $\in\{1,\ldots,m-1\}$ and c (the increment) is a nonnegative integer. This mathematical notation signifies that x_i is the remainder of $(ax_{i-1}+c)$ divided by m. Hence, $x_i \in \{1,\ldots,m-1\}$. Thus, random variable u_i is a uniform 0, 1 variable. Note that

$$x_{i+v} = \left(a^v x_i + \frac{c\left(a^v - 1\right)}{a-1}\right) \text{MOD}\ m\,.$$

Hence, every x_i is completely determined by m, a, c, and x_0. The sequence x_i repeats once it returns to a previously visited value. The period of a generator is the length of a generated steam before it begins to repeat. If $u_0=u_p$ (where $p>0$), then the length p is called the period. The longest possible period for a LCG is m, i.e., m represents the desired number of different values that could be generated for the random numbers. Hence, the modulus m is often taken as a large prime number close to the largest integer directly representable on the computer (i.e., equal or near $2^{31}-1$ for 32-bit computers). If $p=m$, we say that the generator has full period. The required conditions on how to choose m, a, and c so that the corresponding LCG will have full period are known, see Knuth (1997) or Law (2014).

LCGs are sensitive with respect to the parameters, especially the value of a. When $c>0$, the LCGs are called mixed LCGs. When $c=0$,

$$x_i = ax_{i-1} MOD\ m, \quad u_i = \frac{x_i}{m},\ x_0 \in \{1,\ldots,m-1\}\ i > 0.$$

These LCGs are called multiplicative LCGs. Most existing implementations of LCGs are multiplicative LCGs, because in general the value of c does not have a large impact of the quality of an LCG. Note that if $x_i=0$, then all subsequent x_i are identically 0. Thus, the longest possible period for a multiplicative LCG is $m-1$. Furthermore,

$$x_{i+v} = a^v x_i MOD\ m$$

Most experts now recognize that small LCGs with moduli around 2^{31} or so should no longer be used as general-purpose random-number generators. Not only can one exhaust the period in a few minutes on a PC (personal computer), but more importantly the poor structure of the points can dramatically bias simulation results for sample sizes much smaller than the period length.

One way of extending the basic LCG is to combine two or more LCGs through summation. Another way of extending the basic LCG is to use a higher-order recursion. A multiple recursive random number generator (MRG), which goes from integer to integer according to the recursion

$$x_i = \left(a_1 x_{i-1} + \ldots + a_k x_{i-k}\right) \text{MOD}\ m, u_i = \frac{x_i}{m}$$

$$A\ seed\ x_0,\ldots,x_{k-2},x_{k-1} \in \left\{1,\ldots,m-1\right\}$$

where i, k, and m are positive integers, and $a_1,\ldots,a_k \in \{0,1,\ldots,m-1\}$ To increase the efficiency and ease the implementation, the MRG algorithm usually set all but two a_i's to 0. Furthermore, the nonzero a_i should be small. However, L'Ecuyer (2013) points that these conditions are generally in conflict with those required for having a good lattice structure and statistical robustness. See Law (2014) on the lattice structure of pseudorandom numbers. The longest possible period for a MRG is m^k-1 (L'Ecuyer, 1996). Other way of extending the basic LCG is the additive congruential RNG (ACRON). The ACRON sets $a=1$ and replace