A Better Random Number Generator

By Chris Davis

Random numbers are too important to be left to chance. –R.R. Coveyou

Random numbers should not be generated with a method chosen at random. –Donald Knuth

With the adoption of the C-3 Phase 2 Project of the American Academy of Actuaries, it is now required that stochastic scenarios be developed for establishing reserves for variable annuities. This project has led to the idea of Principle-Based Reserving (PBR) where many of the same techniques will be used to set life insurance reserves. A critical item in stochastic testing is the choice of a Random Number Generator (RNG) to develop the random scenarios.

There are two kinds of RNGs—the hardware kind, which are truly random, and the software kind, which are pseudo-random. A hardware random number generator is an electronic device that plugs into a computer and produces genuine random numbers. They are used for generating keys for encryption, winning numbers for lotteries, selecting experimental designs and occasionally for statistical simulations.[1] Hardware generators look to physical processes with random qualities. Examples of these physical processes include decay times of a radioactive material and thermal noise resulting from random electron behavior within a resistor. An example of a hardware generator is a digital noise generator, which can take digital music files and randomly generate a stream of numbers based on the music being played. However, for use in generating random numbers for interest scenarios for cash flow testing or other similar projects, hardware generators can be difficult to use, inefficient for the tasks at hand, too expensive and too slow for routine use in statistical simulations. The remainder of this article is devoted to software generators and the development of a reliable, efficient random number generator.

Most RNGs used in financial applications are of the software kind and produce pseudo-random numbers. "The aim of pseudo random number generators is to implement an imitation of the abstract mathematical concept of mutually independent random variables uniformly distributed over the interval [0,1]."[2] They are called pseudo-random because their output can be predicted since they are based on deterministic computations. The use of pseudo-random numbers in security applications can compromise the application, so hardware RNGs should be used for these types of applications. Similarly, a pseudo-random number generator would not be appropriate for the selection of lottery numbers.

The appointed actuary's results and conclusions must be based on reliable, accurate stochastic testing. The results are only as good as the pseudo-random number generator used in the models. At the same time, an RNG must be easy to use, efficient and cost effective. There have been many articles written about some of the easily available RNGs that do not provide points that are sufficiently random. "The built-in random number generators of proprietary spreadsheets are often not suitable for a large number of simulations or for complex problems."[3] "Do not trust the random number generators provided in popular commercial software such as Excel, Visual Basic, etc., for serious applications. Some of these RNGs give totally wrong answers."[4]

The properties of a good RNG are that it should:[5]

- Provide a very good approximation to the uniform distribution
- Be computationally fast
- Be as close to independent in output in more than just one dimension
- Have a very long period
- Be repeatable, given the starting point

There are some basic mathematical properties to consider when choosing an RNG. The period (p) of an RNG is the number of values generated before the sequence repeats. The period should be as large as reasonably possible. Leading authorities differ on the maximum number (n) of values to be generated. Donald E. Knuth of Stanford University states that n should be at most p/1000. Brian D. Ripley of the University of Oxford has a much more stringent requirement that n is much less than $(p/200)^{0.5}$.

Using Ripley's criteria, an RNG with a period of 16,777,216 (the period of the Rnd function in Visual Basic) should be used only when substantially less than 290 random numbers are required! This is because the discrepancy from true randomness increases as *n* approaches *p*. Substantially less, of course, can be interpreted to be different amounts to different people, but it is obvious that for stochastic modeling purposes, the period will need to be very large. Many commercially available RNGs do not have a sufficiently large period.

Most of the RNGs in use today are linear congruential. That is they use a form of the function:

> $x(i) = (a * x(i - 1) + c) \text{ Mod } m$
> where $0 <= x(i) < m$, a and c are constants and m is a positive integer.
> The final step is to divide $x(i)$ by m to get a number in $(0,1)$.

Linear congruential RNGs are inadequate when more than one stochastic variable is being tested. Example stochastic variables include interest rates, various portfolio performances, lapse rates, mortality rates and policyholder behaviors. The numbers generated by linear congruential RNGs are not randomly spread throughout multi-dimensional space, but instead reside on parallel hyperplanes. Thus many points are not available and randomness is greatly impaired when more than one stochastic variable is present.

One RNG that overcomes these shortcomings is the Mersenne Twister. Marin Mersenne was a French monk and mathematician of the early 17th century who focused on numbers of the form $2^P - 1$, especially prime numbers of this form. Two Japanese mathematicians and computer scientists, Makoto Matsumoto and Takuji Nishimura, developed the Mersenne Twister in 1997. The period for this algorithm is $2^{19937}-1$, which is the prime number $4.315 * 10^{6001}$. It uses an algorithm that is a twisted generalized feedback shifter. The "twist" is a transformation, which assures equidistribution of the generated numbers in 623 dimensions.[6] There is much less correlation between successive values. It is a fast and efficient algorithm. As with all pseudorandom number generators, the Mersenne Twister should not be used in areas of cryptology. But it is a reliable generator for use in stochastic modeling.

To be considered a generator of worth, it is generally subjected to a series of statistical tests. One of the most stringent sets of tests is the Diehard Battery of Randomness Tests, developed by George Masaglia at Florida State University and first published in 1995[7]. This is a battery of 15 statistical tests (some with very colorful, descriptive names) used to measure the quality of a set of random numbers. The values generated from the Mersenne Twister pass the Diehard tests.

The Mersenne Twister is quickly becoming the RNG of choice for many applications. The prepackaged scenarios developed for the Academy's C-3 Phase 2 work used a robust RNG which was a variant of the Mersenne Twister. There is substantial discussion in the statistical literature that emphasizes the use of a quality random number generator. A thorough audit of stochastic testing should address the underlying RNG. "An expensive house built on shaky foundations is a shaky house. This applies to expensive simulations as well."[8] At Lewis & Ellis, we have implemented the Mersenne Twister as an Excel add-in and as a DLL.

There is substantial discussion in the statistical literature that emphasizes the use of a quality random number generator.

*   *   *   *   *   *

[1] Robert Davies, "Hardware Random Number Generators," October 14, 2000 accessed Dec. 29, 2005 from <http://www.robertnz.net/hwrng.htm>.

[2] Brian D. Ripley, "Thoughts on pseudorandom number generators," Journal of Computational and Applied Mathematics, 31, 153–163.

[3] Mary Hardy FSA FIA PhD, *Investment Guarantees: Modeling and Risk Management for Equity-Linked Life Insurance*, Mary Hardy, (Hoboken: Wiley, 2003) 104.

[4] Pierre L'Ecuyer, "Software for Uniform Random Number Generation: Distinguishing the Good and the Bad," accessed Jan. 16, 2006 from <http://www.iro.umontreal.ca/~lecuyer/myftp/papers/wsc01rng.pdf>.

[5] Ibid.

[6] "Mersenne Twister," *Wikipedia*, accessed Jan. 16, 2006 from <http://en.wikipedia.org/wiki/Mersenne_twister>.

[7] Ibid.

[8] Pierre L'Ecuyer, "Software for Uniform Random Number Generation: Distinguishing the Good and the Bad," accessed Jan. 16, 2006 from *<http://www.iro.umontreal.ca/~lecuyer/myftp/papers/wsc01rng.pdf>*. ●

Christopher H. Davis, FSA, MAAA, is vice president and principal with Lewis & Ellis, Inc. in Overland Park, Kan. He can be reached at cdavis@lewisellis.com.