

Stochastic Simulation

Michaelmas Term 2002

Dr. Gesine Reinert

Organization of the class

This class will take place Wednesdays 12-1, from week 1 to week 8, in the Department of Statistics. In addition there will be a practical class on

Tuesday, November 19 (week 6), in the Computer lab in the department

Lecture notes and a problem sheet will be handed out in the beginning of every second week. For computer exercises, we will use S-PLUS. Note that lectures may differ slightly from these lecture notes.

Recommended reading

1. J.M. HAMMERSLEY AND D.C. HANDSCOMB (1964). *Monte Carlo Methods*. Methuen.
2. A.M. LAW AND W.D. KELTON (1999). *Simulation Modeling and Analysis*. Third edition, McGraw-Hill.
3. B.J.T. MORGAN (1984). *Elements of Simulation*. Chapman and Hall.
4. B.D. RIPLEY (1987). *Stochastic Simulation*. Wiley.
5. S.M. ROSS (1996). *Simulation*. Second edition. Academic Press.

1 Introduction

Over the last 50 years, simulation has become an important tool in applied mathematics, in computer science, and in engineering. Often a real-life problem is too complex for a closed-form solution; think of the aerodynamics of an airplane, or of weather systems. In principle, these complex deterministic systems could be described completely, for example by differential equations,

but these are too large and/or too complicated to be solvable exactly. Hence one would resort to simulation to understand the underlying behaviour.

In addition, there are complex problems that involve randomness and hence are not solvable deterministically at all. Among these would be, for example, internet traffic, and the spread of disease in a population. In this class we will restrict our attention to this type of problems, namely those that have an element of randomness. Some uses of simulation would be

- exploratory modeling of the behaviour of complex stochastic systems, such as queuing systems, spatial spread of epidemics
- to gain insight in statistical behaviour, such as simulating from a binomial distribution to illustrate the central limit theorem
- intractable deterministic problems in analysis sometimes have a solution involving stochastic processes, and these could be studied by simulation
- examination of the properties of statistical procedures, such as estimating the power of tests, assessing robustness of procedures
- statistical inference: determine the null distribution of a statistic

Advantages of simulations are that we could try different scenarios easily, that we could analyze systems of almost arbitrary complexity, and that we could visualize processes. Disadvantages are that a simulation might be slow, and that optimization of the algorithms can be hard. Moreover, rare event simulation is extremely difficult. Large scale modeling is still very hard and time consuming.

2 Pseudo-random number generators

For any simulation of stochastic systems, we need to be able to simulate “randomness”, that is, to simulate observations that look as if they came from a prespecified probability distribution. One way of achieving this would be for example to flip a fair coin repeatedly, to roll a fair die repeatedly, or to draw a card from a deck of cards repeatedly.

For example, if you wanted to determine the probability that with 4 cards drawn at random from a deck of 52 cards, exactly two of them are aces, you could simulate this event by shuffling repeatedly, dealing 4 cards, and recording whether or not they contained two aces.

In simulations often we need many random numbers, so this is not feasible. (Think of assessing insurance risks - these are typically rare events.) On the other hand, computer algorithms are deterministic in nature, so it is not obvious how to generate random numbers in a computer. Of course statistical software comes with pseudo-random number generators, but this class is partly designed to look behind that curtain. In particular we will see that these pseudo-random generators have its flaws, and these flaws are important to know when setting up a simulation.

2.1 Generating from the uniform distribution

The best distribution to start with is the uniform distribution on $[0, 1]$, denoted by $\mathcal{U}([0, 1])$. Ideally, a pseudo-random number generator (RNG) for $\mathcal{U}([0, 1])$ would

- provide a very good approximation to $\mathcal{U}([0, 1])$ marginally
- have very close to independent output in a moderate number of dimensions
- be repeatable from a specified starting point
- be fast
- not repeat itself too often
- be portable (for different operating systems)
- be analyzable
- in cryptography often it is desired that the RNG is unpredictable.

Virtually all random number generators are based on the following idea. We have a finite set E and a function $f : E \rightarrow E$. Given an initial value X_0 , the generated sequence is

$$X_0, X_1 = f(X_0), X_2 = f(X_1) = f^2(X_0), \dots$$

The three most common types of pseudo-random number generators are the so-called congruential generators, the shift-register generators, and lagged-Fibonacci generators, with variants.

2.1.1 Congruential generators

The (*first-order*) *congruential generators* were introduced by Lehmer (1951). They are of the form

$$X_n = aX_{n-1} + b \pmod{M}$$

so that $X_n \in [0, M - 1]$. If $b = 0$ this is called a *multiplicative* generator, if $b \neq 0$ it is called a *mixed* generator. We then put

$$U_i = \frac{X_i}{M}.$$

The starting point X_0 is called the *seed*. The seed is often chosen by the programmer, or using the internal clock of the computer. Ideally, the seed should be chosen at random from $\mathcal{U}([0, 1])$.

A convenient choice for M would equal the computer's word length, since then division by M is quite efficient. However, to reduce periodicity, M should be large.

We also want the sequence not to repeat itself too often. A congruential generator yields a periodic sequence with period k , say. For mixed congruential generators, $k \leq M$. If $k = M$, the generator is said to have *full period*. For multiplicative congruential generators, $k \leq M - 1$, since 0 repeats itself indefinitely. If $k = M - 1$, the multiplicative generator is called *maximal*.

In Hammersley and Hanscomb (1964), p.28 (see also Ripley (1987)) the following can be found.

Proposition 1 *For a mixed congruential generator, the full period of M can always be achieved provided that*

1. b and M have no common divisor
2. $a \equiv 1 \pmod{p}$ for every prime factor of M
3. $a \equiv 1 \pmod{4}$ if M is a multiple of 4.

For multiplicative generators, we can use the following fact (Ripley (1987), p.21).

Proposition 2 *A multiplicative generator has period $M - 1$ only if M is prime. Then the period divides $M - 1$, and is $M - 1$ if and only if a is a primitive root, that is, $a \neq 0$ and $a^{\frac{M-1}{p}} \not\equiv 1 \pmod{M}$ for each prime factor p of $M - 1$.*

Popular examples would be $M = 2^{32}$, $a = 69,069$, $b = 1$, or the *Learmonth-Lewis generator* $M = 2^{31} - 1$, $a = 7^5 = 16,807$, $b = 0$. The S-PLUS function `congrval` uses $a = 69,069$, $b = 0$, $M = 2^{32}$. A rather unpopular example is $M = 2^{31}$, $a = 2^{16} + 3$, $c = 0$, see Ripley (1987), p.23-24.

Note that successive numbers generated by these methods would always be correlated. Depending on the choice of parameters the correlation could be quite strong. Successive values all lie on a lattice, see Marsaglia (1968), and the goal is to choose a such that the lattice is rather evenly spread, and that the points are not too far apart. Shuffling can also help.

2.1.2 Shift-register generators

For *shift-register* generators, the set E is the set of binary k -vectors $X = (X_1, X_2, \dots, X_k)$, and f is a linear transformation $f(x) = xT$, with T a binary $k \times k$ matrix and all calculations being carried out *mod*2. This is convenient as it corresponds to the exclusive “or” (EOR). An example is

$$X_n = a_n X_{n-1} + a_2 X_{n-2} + \dots + a_k X_{n-k} \pmod{2},$$

where $a_1, \dots, a_k \in \{0, 1\}$. Then put

$$U_n = 0.X_{n-1} \dots X_{n-k}.$$

Very customary is to use

$$X_n = X_{n-p} + X_{n-q} \pmod{2};$$

here, $1 \leq q < p$. Choices of (p, q) that give the maximal period $2^p - 1$ are given in Ripley (1987), p.27.

The *Tausworthe* generator, suggested by Tausworthe (1965), uses L -bit binary fractions taken t apart:

$$U_n = \sum_{s=1}^L 2^{-s} X_{nt+s} = 0.X_{nt+1}X_{nt+2} \dots X_{nt+L},$$

that is, L -bit binary fractions taken t apart. Here, $0.X_{nt+1}X_{nt+2} \dots X_{nt+L}$ is a binary representation. The parameter $t \geq L$ is called the *decimation*, and if t and $2^p - 1$ are relatively prime, then the decimation is said to be *proper* and the sequence attains its full period.

Typical values would be $p = 33, q = 13, t = L = 32$, having period $2^{33} - 1$.

Recently the Mersenne Twister, developed by Matsumoto and Nishimura (1998), has become rather popular. It is based on a matrix version of a shift-register generator.

2.1.3 Lagged-Fibonacci generators

Here, E is the set of r -vectors with elements in some finite set. The function f is defined by

$$f(X_1, X_2, \dots, X_r) = (X_2, X_3, X_4, \dots, X_r, X_1 + X_{r+1-s}) \pmod{M}.$$

An example is

$$X_n = X_{n-607} - X_{n-243} \pmod{2^{32}}.$$

The advantage of lagged-Fibonacci generator is that their period is considerably larger than the period of congruential generators. The above generator, for example, has period about 2^{607+32} .

L'Ecuyer (1999) showed that the combined linear multiple recursive generator

$$\begin{aligned} X_{1,n} &= (a_{1,1}X_{1,n-1} + \dots + a_{1,k}X_{1,n-k}) \pmod{m_1} \\ X_{2,n} &= (a_{2,1}X_{2,n-1} + \dots + a_{2,k}X_{2,n-k}) \pmod{m_2} \\ U_n &= \left(\frac{X_{1,n}}{m_1} - \frac{X_{2,n}}{m_2} \right) \pmod{1} \end{aligned}$$

has good properties for

$$k = 3$$

$$\begin{aligned}
m_1 &= 2^{32} - 209 \\
m_2 &= 2^{32} - 22,853 \\
(a_{1,1}, a_{1,2}, a_{1,3}) &= (0; 1,403,480; -810,728) \\
(a_{2,1}, a_{2,2}, a_{2,3}) &= (527,612; 0; -1,370,589)
\end{aligned}$$

It has two main cycles of length $\approx 2^{191}$; the lattice test is satisfactory for tuples of length at most 48.

Lagged-Fibonacci Generators have been refined by *subtract-with-borrow* generators, see Marsaglia and Zaman (1991). These are of the form

$$X_n = X_{n-s} - X_{n-r} - b \pmod{M}.$$

Here, b is a “borrow” flag. The iterating function here is

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

These have much longer periods, for example, with $M = 32, s = 607, r = 243$ the above generator has period about $2^{607 \times 32}$. Unfortunately, also this generator has the problem of falling mainly on the planes; indeed, Tezuka, L’Ecuyer and Couture (1993) showed that all triples (X_n, X_{n-s}, X_{n-r}) lie on only two planes in $[0, 1]^3$.

There is no uniformly best pseudo-random number generator available. The Mersenne Twister seems to display good properties. Combining generators is also a good choice. For example, one could add the binary output of two generators mod 2 and use that as a new random number. Or one could have two random number generators and a third one to switch between the two. Another option is shuffling the output. There are nonlinear generators available, displaying more favorable statistical properties, but unfortunately they are still rather slow, see L’Ecuyer (1998, 2002).

2.2 Testing randomness

A good RNG should produce an output which does not differ significantly from that of a (memoryless and fair) monkey hitting keys on a numeric keyboard. Given a pseudo-random number generator, statistical tests are advisable. Here are some possibilities.

2.2.1 Testing for independence

The *gap test* is based on the following observation. Choose two numbers, $\alpha \leq \beta \in [0, 1]$, say, and record the lengths of the subsequence lying between occurrences of the same sequence within $[\alpha, \beta]$. If a sequence X_1, X_2, \dots, X_n was chosen uniformly and independently, then the distribution of the gap length K should be geometric with probability of success

$$P(\alpha \leq U \leq \beta) = \beta - \alpha,$$

where U denotes a $\mathcal{U}([0, 1])$ random variable. Under the independence assumption, successive gap lengths are independent, and a Chisquare test for independence can be used for testing whether the gap lengths are indeed independent. This assumes that $U \sim \mathcal{U}([0, 1])$; indeed the gap test can be used for other distributions as well, with a modified probability of success.

The *run test* follows a related idea. Record the length of *runs*, these are monotonically increasing subsequences of X_1, X_2, \dots, X_n . For example, $(3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5) = 3|14|159|26|5|35$ has 2 runs of length 1, 3 runs of length 2, and one run of length 3. Under the null hypothesis that the observations are independent, the run lengths are independent, with expectation

$$\mathbf{E}(\text{number of runs with length } k) = \frac{(n+1)k}{(k+1)!} - \frac{k-1}{k!}, \quad k = 1, \dots, n.$$

A *permutation test* would divide X_1, X_2, \dots, X_n into blocks of length t , say; $(X_1, \dots, X_t), (X_{t+1}, \dots, X_{2t}),$ and so on. Under the null hypothesis of independence, all $t!$ different orderings should be equally likely, and a Chisquare test can be used to test this.

There are also tests based on *return time analysis*, see (*Wegenkittl (1999)*)

2.2.2 Testing for the distribution

For this we could easily use the Kolmogorov-Smirnov test; if

$$F_n(x) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}(X_i \leq x)$$

denotes the empirical distribution function of the observations, and F is the cumulative distribution function of the true distribution (which is assumed

to be continuous), then the Kolmogorov-Smirnov statistic

$$\sup_x |F_n(x) - F(x)|$$

should be small if the distribution is indeed uniform, and larger otherwise. For $\mathcal{U}([0, 1])$, we have the special case $F(x) = x$, $0 \leq x \leq 1$. The distribution of this test statistic is not easy to determine; it can be shown (see Shorack (2000), p. 316) that

$$\lim_{n \rightarrow \infty} \mathbf{P}(\sqrt{n} \sup_x |F_n(x) - x| > \lambda) = 2 \sum_{k=1}^{\infty} (-1)^{k+1} \exp(-2k^2 \lambda^2).$$

Instead, one could also bin the observations and use a Chisquare test. Make a histogram of d equal size bins; then, under H_0 , the probability for each bin is $\frac{1}{d}$. Use a Chisquare test with $d - 1$ degrees of freedom.

Another alternative is the *Maximum-of-t Test*. Divide the sequence into n groups of t elements each, and calculate the maximum of each group. Under H_0 these maxima should follow the cumulative distribution function $F(x) = x^t$, $0 \leq x \leq 1$. Use a Kolmogorov-Smirnov test.

For testing uniformity of k -vectors, under the null hypothesis of uniformity, the k -vectors $(U_1, \dots, U_k), (U_{k+1}, \dots, U_{2k}) \dots$ should be independent and identically uniformly distributed on the cube $[0, 1]^k$. Thus divide $[0, 1]^k$ into d subcubes of same size, count the number of observations in each cube, and use a Chisquare test.

For testing for independence when uniformity is assumed, there is the *Serial Correlation Test*. Let U_0, U_1, \dots, U_{n-1} and V_0, V_1, \dots, V_{n-1} be two series. Then a correlation coefficient between the two series can be defined by

$$C = \frac{n \sum_j U_j V_j - \sum_j U_j \sum_j V_j}{\sqrt{(n \sum_j U_j^2 - (\sum_j U_j)^2) (n \sum_j V_j^2 - (\sum_j V_j)^2)}}.$$

Let U_0, U_1, \dots, U_{n-1} be uniform, and put $V_j = U_{(j+1) \bmod n}, j = 0, \dots, n - 1$. Then

$$C = \frac{n(U_0 U_1 + U_1 U_2 + \dots + U_{n-1} U_0) - (U_0 + U_1 + \dots + U_{n-1})^2}{n(U_0^2 + U_1^2 + \dots + U_{n-1}^2) - (U_0 + U_1 + \dots + U_{n-1})^2}.$$

The exact distribution for C is not known; empirically “good” values of C lie between $\mu_n - 2\sigma_n$ and $\mu_n + 2\sigma_n$ 95% of the time. Here, $\mu_n = -\frac{1}{n-1}$, $\sigma_n = \frac{1}{n-1}\sqrt{\frac{n(n-3)}{n+1}}$, for $n > 2$.

Recently the *Collision test* has been advertised, (*Knuth (1997)*). Cut $[0, 1)$ into k equal intervals, generate n points independently in $[0, 1)$, and let C = number of times a point falls in a box that already has a point in it. For large k , C is approximately Poisson ($\frac{n^2}{2k}$). Choose $k \approx$ equal to the period, and test for Poisson. This can be generalized to higher dimensions.

A variant of this is the *Birthday spacings test* (*L’Ecuyer and Simard (2001)*). Cut $[0, 1)$ into k equal intervals, and number these boxes in natural order. Generate n points independently in $[0, 1)$, and let $I_1 \leq I_2 \leq \dots \leq I_n$ be the intervals where points fell. Consider the spacings

$$S_j = I_{j+1} - I_j, \quad j = 1, \dots, n-1.$$

Let Y be the number of collisions between these spacings (that is, $S_{(j)} = S_{(j+1)}$). This corresponds to the birthday problem with n people and year with k days. Under H_0 , Y is approximately Poisson(λ), if $\lambda = \frac{n^3}{4k}$ is small.

In (*L’Ecuyer, Simard and Wegenkittl (2002)*) these tests are studied in a more general framework. Partition $[0, 1)$ into d equal segments. This generates a partition of $[0, 1)^t$ into $k = d^t$ cubes of equal size. Generate U_0, \dots, U_{nt-1} random numbers, put

$$V_{ti} = (U_{ti}, \dots, U_{ti+t-1}), \quad i = 0, \dots, k-1,$$

and let X_j be the number of these points falling into cube j , $j = 0, \dots, k-1$. Let

$$\lambda = \frac{n}{k}$$

denote the average number of points per cube under the null hypothesis that the data are i.i.d. $\mathcal{U}([0, 1])$. Pearson’s Chi-square statistic is

$$X^2 = \sum_{j=0}^{k-1} \frac{(X_j - \lambda)^2}{\lambda};$$

under the null hypothesis, X^2 is approximately $\chi_{(k-1)}^2$ distributed, when $\lambda \geq 5$, say. This can be generalized to test statistics

$$Y = \sum_{j=0}^{k-1} f_{n,k}(X_j),$$

where $f_{n,k}$ is a real valued function. For example, the function $f_{n,k}(x) = \frac{(x-\lambda)^2}{\lambda}$ gives Pearson's Chi-square statistics, the function $f_{n,k}(x) = \mathbf{1}[x = b]$ gives the number of cells with exactly b points, and $f_{n,k}(x) = (x-1)\mathbf{1}[x > 1]$ gives the number of collisions. It is calculated in (*L'Ecuyer, Simard and Wegenkittl (2002)*) that, under the null hypothesis,

$$\begin{aligned}
EY &= k\mu = \sum_{x=0}^n \binom{n}{x} \frac{(k-1)^{n-x}}{k^{n-1}} f(x) \\
VarY &= \sum_{x=0}^n \binom{n}{x} \frac{(k-1)^{n-x}}{k^{n-1}} (f(x) - \mu)^2 \\
&\quad + \sum_{x=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{x} \binom{n-x}{x} \frac{(k-1)(k-2)^{n-2x}}{k^{n-1}} (f(x) - \mu)^2 \\
&\quad + \sum_{x=0}^n \sum_{y=0}^{\min(n-x, x-1)} \binom{n}{x} \binom{n-x}{y} \frac{(k-1)(k-2)^{n-x-y}}{k^{n-1}} \\
&\quad \cdot (f(x) - \mu)(f(y) - \mu).
\end{aligned}$$

There are two asymptotic regimes to distinguish: the *sparse* case where λ is small, and the *dense* case where $\lambda > 1$. In the sparse case, the count statistic and the collision statistic will be approximately Poisson distributed, whereas in the dense case, they will be approximately normal distributed.

These tests can be extended to overlapping vectors.

Due to the periodic nature of many RNGs, for large sample sizes the null hypothesis is likely to be rejected. (*L'Ecuyer, Simard and Wegenkittl (2002)*) also give an empirical evaluation for RNGs. They conclude that all linear congruential generators and Lagged-Fibonacci shift register generators fail the above tests as soon as the sample size exceeds a few times the square root of their period length, regardless of the choice of their parameters. Thus they advice against using RNGs with small periods (*small* here meaning less than 2^{50}).

Another type of tests uses the generated numbers to approximate something that is known already, and see how well it does. An example for this is the *Monte Carlo Value for π* . Each successive sequence is used as X and Y co-ordinates within a square. If distance from zero of a point is smaller

than the radius of a circle inscribed within the square, a hit is recorded. The percentage of hits approximates (very slowly) π .

Marsaglia compiled a battery of tests of randomness called DIEHARD, to be found at <http://stat.fsu.edu/pub/diehard/>. A newer version is at <http://www.helsbreth.org/random/diehard.html>.

Another battery called ENT has been developed by John Walker, see <http://www.fourmilab.ch/random/>.

Further reading

1. D.E. KNUTH (1997). *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. 3rd ed., Addison-Wesley; Reading, Mass.
2. P. L'ECUYER (2002). Random Numbers. In *the International Encyclopedia of the Social and Behavioral Sciences*, N. J. Smelser and Paul B. Baltes Eds., Pergamon, Oxford, 12735–12738. (A short and easy introduction to random number generation.)
3. P. L'ECUYER (2001). Software for uniform random number generation: Distinguishing the good and the bad. *Proceedings of the 2001 Winter Simulation Conference*, IEEE Press, 95–105.
4. P. L'ECUYER (1999). Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, **47** (1), 159–164.
5. P. L'ECUYER (1998). Uniform random number generators. *Proceedings of the 1998 Winter Simulation Conference*, IEEE Press, 579–586.
6. P. L'ECUYER, R. SIMARD, AND S. WEGENKITTL (2002). Sparse serial tests of uniformity for random number generators. *SIAM Journal of Scientific Computing*. To appear.
7. P. L'ECUYER AND R. SIMARD (2001). On the performance of birthday spacing tests with certain families of random number generators. *Mathematics and Computers in Simulation* **55**, 131–137.

8. D.H. LEHMER (1951). Mathematical methods in large-scale computing units. *Proceedings of the Second Symposium on Large-Scale Digital Calculating Machinery*. Harvard University Press, Cambridge, MA, 141–146.
9. G. MARSAGLIA (1968). Random numbers fall mainly in the planes. *Proc. Nat. Acad. Sci. USA*, **61**, 25–28.
10. G. MARSAGLIA AND A. ZAMAN (1991). A new class of random number generators. *Ann. Appl. Probab.* **1**, 462–480.
11. M. MATSUMOTO AND T. NISHIMURA (1998). Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom generator. *ACM Transactions on Modeling and Computer Simulations: Special Issue on Uniform Random Number Generation*.
12. B.D. RIPLEY (1990). Thoughts on pseudorandom number generators. *J. Comput. Appl. Math.* **31**, 153–163.
13. G.R. SHORACK (2000). *Probability for Statisticians*. Springer.
14. R.C. TAUSWORTHE (1965). Random numbers generated by linear recurrence modulo two. *Math. Comp.* **19**, 201–209.
15. S. TEZUKA (1995). *Uniform Random Numbers: Theory and Practice*. Kluwer; Norwell, Mass.
16. S. TEZUKA, P. L’ECUYER, AND R. COUTURE (1993). On the lattice structure of the add-with-carry and subtract-with-borrow random number generators. *ACM Transactions on Modeling and Computer Simulations* **3**, 315–331.
17. WEGENKITTL, S. (1999). Monkeys, gambling, and return times: Assessing Pseudorandomness. *Proceedings of the 1999 Winter Simulation Conference*, IEEE Press, 625–631.

Some useful URLs

Pierre L’Ecuyer’s home page
<http://www.iro.umontreal.ca/~lecuyer/>

Mersenne Twister home page
<http://www.math.keio.ac.jp/~matumoto/emt.html>

Marsaglia's random number CDRom at
<http://stat.fsu.edu/~geo>

P-Lab
<http://random.mat.sbg.ac.at/>