Journal of Simulation (2007) 1, 131–135

www.palgrave-journals.com/jos

Stream correlations in multiple recursive and congruential generators

R Davies[1]* and RJ Brooks[2]

[1]*University of Warwick, Coventry, UK; and* [2]*Lancaster University, Lancaster, UK*

The use of common random numbers is a method of variance reduction usually implemented by allocating different but repeatable random number streams to different distributions in a simulation. Linear or multiplicative congruential generators with linearly related seeds for the streams are commonly used. We show that there is a simple mathematical relationship between the *i*th number in streams of multiple recursive or congruential generators and that streams produced from linearly related seeds have serious undesirable correlations.
Journal of Simulation (2007) **1,** 131–135. doi:10.1057/palgrave.jos.4250013

Keywords: random numbers; variance reduction; seeds; streams; correlation

1. Introduction

The random sampling of values from a probability distribution is an essential part of most discrete event simulation models. The basis of this process is usually a mathematical algorithm called a pseudo-random number generator (RNG), which produces a sequence (or stream) of numbers 'randomly' sampled from the Uniform (0,1) distribution. A variety of RNGs have been proposed in the literature (eg see the review by L'Ecuyer, 1990 or the discussion in Press *et al*, 1992). Three of the most commonly used methods are: the linear congruential generator (LCG), the multiplicative congruential generator (MCG) and the multiple recursive generator (MRG). Each of these methods generates numbers from previously generated numbers, using a mathematical formula, and the sequence starts with a predetermined initial number or set of numbers, called seeds. The sequence of numbers is referred to as a stream.

An important benefit of RNGs in simulation is the ability to have control over the random numbers used as this allows experiments to be repeated exactly (eg Brooks and Robinson, 2001; Pidd, 2004). It also enables certain variance reduction methods to be employed such as the common random numbers method and the use of antithetic random variates (discussed, eg in Law 2007 and in Pidd, 2004). The aim of variance reduction methods is to identify the effect of changes in the experimental variables more clearly when comparing different scenarios, and therefore to reduce the number of replications needed.

This paper identifies a potential pitfall when applying the common random numbers method to a simulation using an LCG, MCG or MRG.

**Correspondence: R Davies, Warwick Business School, University of Warwick, Coventry CV4 7AL, UK.*
E-mail: ruth.davies@wbs.ac.uk

2. Random number generators

The LCG, which was first described by Lehmer (1951), uses the following algorithm to produce a sequence of integers x_i using the recurrence relation:

$$x_i = (ax_{i-1} + c) \bmod m \tag{1}$$

where a, c and m are positive integer constants. The integers x_i are therefore each between 0 and $m-1$ and the stream of pseudo-random numbers between 0 and 1 output by the LCG are obtained simply by dividing each integer x_i by m. The first integer, x_0, is an input parameter called the seed and can be any integer between 0 and $m-1$. The MCG uses the same method as the LCG except that the constant c in Equation 1 equals zero. The seed 0 is not allowed in this case as it would then be repeated at every value of x_i.

A deterministic algorithm cannot be truly random and the LCG and MCG have two well-known weaknesses. Firstly, they do not sample from all numbers between 0 and 1 but only from the finite set $\{0, 1/m, 2/m, \ldots, (m-1)/m\}$ (excluding 0 for the MCG). Secondly, the numbers produced for a given seed will be a transient sequence followed by a repeating cycle, with the length until the first repeat (transient length plus cycle length) being at most m for the LCG and $m-1$ for the MCG. These generators also inherently contain some serial correlations. For example, a and c are usually much smaller than m and consequently an extremely small number will always be followed by a small number.

The weaknesses of the LCG and MCG can be largely mitigated by a very careful choice of the values for the constants a, c, and m. For example, making m very large means that the set of numbers sampled from are very close together and so give a good enough coverage of the interval

[0, 1] for most applications. If certain conditions are satisfied for a, c, and m, then the generator will produce a single cycle of maximal length (m for the LCG or $m-1$ for the MCG) and is then known as a full period generator (Law, 2007). A full period generator with a large m will therefore have a large cycle length. The values often recommended for the constants are those originally suggested for an MCG by Lewis *et al* (1969), where $a = 16\,807$, $c = 0$ and $m = 2^{31}-1 = 2\,147\,483\,647$. An additional consideration is the coding of congruential generators, since this is not as straightforward as it might appear, with care being required to avoid integer overflow in the calculations (Park and Miller, 1988).

With a good choice of constants, LCGs and MCGs are generally considered to perform well and to be suitable for most simulation applications. This, combined with their simplicity, has resulted in their widespread use in simulation.

Both LCGs and MCGs are specific cases of the MRG:

$$x_i = (a_1 x_{i-1} + a_2 x_{i-2} + \cdots + a_r x_{i-r}) \bmod m \quad (2)$$

No constant c is required because this can always be eliminated using the equation for x_{i-1}. For example, rearranging the LCG Equation (1) for x_{i-1} gives

$$c = (x_{i-1} - a x_{i-2}) \bmod m \quad (3)$$

Substituting into (1),

$$x_i = ((a + 1)x_{i-1} - a x_{i-2}) \bmod m \quad (4)$$

Using x_0 and x_1 as the two initial seeds for this MRG, therefore, produces the same sequence of numbers as the LCG. L'Ecuyer (1994) applied 10 statistical tests to a variety of generators and found that most 'failed spectacularly at least one of the tests'. However, one generator that did not do this was an MRG with $r = 5$, $a_1 = 107\,374\,182$, $a_5 = 104\,480$, $a_2 = a_3 = a_4 = 0$, and $m = 2^{31}-1$.

An MRG $x_i = (a_1 x_{i-1} + a_2 x_{i-2} + \cdots + a_r x_{i-r}) \bmod m$ (Equation 2) with r terms can be represented in matrix notation by:

$$X_i = A X_{i-1} \bmod m \quad (5)$$

where $X_i = (x_i, x_{i-1}, \ldots, x_{i-r+1})^T$ and $X_0 = (x_0, x_{-1}, \ldots, x_{-r+1})^T$ is the initial seed (T denotes transpose, ie these are column vectors) and A is the $r \times r$ matrix $[A_{ij}]$ with

$$A_{ij} = \begin{cases} a_j & \text{if } i = 1 \\ 1 & \text{if } i > 1 \text{ and } j = i - 1 \\ 0 & \text{otherwise} \end{cases}$$

Therefore,

$$X_i = A^i X_0 \bmod m \quad (6)$$

3. Common random numbers method and multiple streams

The common random numbers method is usually applied by allocating a different random number stream to each different sampling distribution used in the simulation. Each stream must be generated from a known input seed so that the experiment can be repeated exactly if required. In practice, linearly related seeds are often chosen (eg 1, 10 001, 20 001, etc).

The use of common random number streams is a popular method of variance reduction. Experimentation in simulation consists of changing one or more parameter value (or even the system structure) and comparing the results. There is often one base scenario representing the current situation and other, comparative scenarios, representing possible changes that may be made to the real system. The parameter values may, for example, be resources, probability values, or means and standard deviations of distributions.

Scenarios can be compared by calculating the differences in the output variable(s), with the estimate of the mean difference often being an important statistic. For each scenario, several values of the output variable are obtained, either from analysis of one long simulation run (eg using the batch means method) or, more usually, by multiple replications (Law, 2007). By following the same procedure (eg the same number of replications) for each scenario, the differences in the corresponding pairs of values (eg from the ith replication) can be used to compare the scenarios. If Y_i are the output values from the simulations of the one scenario, and Z_i are the equivalent output values from the simulations of the comparative scenario, the values for the difference are $Z_i - Y_i$, and

$$\mathrm{Var}(Z_i - Y_i) = \mathrm{Var}(Y_i) + \mathrm{Var}(Z_i) - 2\mathrm{Cov}(Y_i, Z_i) \quad (7)$$

If Y_i and Z_i are independent, then the covariance is zero and the variance of the difference is equal to the sum of the variances of the two sets of values. This can be extremely large necessitating long simulation runs or a large number of replications to get a sufficiently precise estimate of the mean difference (ie a small enough confidence interval). The principle of the common random numbers method is to introduce a high degree of correlation, and hence covariance, between the Y_i and Z_i values in order to reduce the variance of the difference. This should reduce the number of replications required and hence reduce the computing time required for the experiments. This can be very important for large or complex models with a significant run time.

The idea of the common random numbers method is to compare the scenarios in the same or, at least, very similar circumstances. For example, where the entities sample decisions and times at certain points in a particular replication, the same random numbers are used as far as possible for each scenario. Differences in output arise, therefore, from parameter changes alone; this gives rise to a high degree of correlation between the output values

measured. In order to achieve this, each distribution is usually sampled using a separate random number stream, with the same streams being used between scenarios. A new replication is achieved by using a completely different set of streams, but again using the same streams between scenarios.

Consider a simple example of a discrete event simulation of a simple FIFO (first-in first-out) one-stage queueing situation in which entities arrive, join a queue, are served and leave. Parameters that might be changed in an experiment are the mean inter-arrival time, the mean service time and the number of servers. Other possible experiments are to change other parameters of the distributions (eg the standard deviation), the type of the distributions or the queue discipline. The entities will progress through the system in the same order in each experiment, except when the queue discipline is changed. Suppose that we are interested in the effect of changing the average service time on the average queue length, the average waiting time and the average server busy time. To apply the common random numbers method, separate random number streams would be allocated to the generation of inter-arrival times and service times and the same seeds used in replication 1 for both scenarios. Then the customers will arrive at exactly the same time in the two scenarios. The random numbers used for the service time will also be the same and so the only difference between the scenarios will be that the service times differ by the experimental factor of the change in the mean. Effectively the two replications will represent the same conditions (the same day) with the only difference being the speed of the server. The differences in the results will be due to the change in the mean service time rather than a combination of a change in mean and a change in the random numbers used. Replication 2 would be obtained by changing the seeds from replication 1, but again both scenarios would use the same seeds.

The same principle can be applied to more complex simulations and this method can be very effective in systems where the order of processing is the same in the two scenarios. Law (2007) illustrates this with the analysis of the average customer waiting time in such a one-stage FIFO queueing system, comparing a one server system with a two server system of the same utilization (comparison of an M/M/1 and an M/M/2 queue). The common random numbers method produced a reduction in variance of 99%. However, in simulations where the routes or order of the entities is different in the two scenarios, the method is less effective unless precautions are taken to link random numbers to individual entities.

When using different streams for different sources of variability, the streams must be independent of each other so that there is no unintended correlation between different parts of the simulation. However, the next section shows that these correlations can be present when using a recursive generator.

4. Relationships between streams

Suppose there are n streams of an MRG where the ith value from the jth stream is denoted $X_{i,j}$ $(j=1,2,\ldots,n)$. We will consider the relationships between any two of these streams $X_{i,j}$ and $X_{i,k}$.

Previous work has been carried out by Anderson and Titterington (1993) and Savage *et al* (1994) on a pair of streams from an MCG (although the results can probably be extended to the general MRG case), which we will denote by $x_{i,j}$ and $x_{i,k}$. Both groups of authors considered the relationship between all the corresponding values (ie the set of pairs of values $\{(x_{i,j}, x_{i,k})$: for all $i\}$), when the seeds are related by a rational number b_1/b_2 where b_1 and b_2 are positive integers. They showed that if:

$$b_1 x_{0,j} \bmod m = b_2 x_{0,k} \bmod m \tag{8}$$

then this relationship is maintained in the corresponding ith values,

$$b_1 x_{i,j} \bmod m = b_2 x_{i,k} \bmod m \tag{9}$$

The key result that they derived from this is that the set of points $(x_{i,j}, x_{i,k})$ will lie on at most b_1+b_2-1 equally spaced parallel lines. With a truly random pair of streams, the points would fill the unit square. Therefore, a poor choice of seeds can produce a strong linear pattern and a high correlation between the streams.

4.1. Relationship between the ith value of the streams

In our analysis, we consider the general relationship between the ith value of the n streams of an MRG and how this is affected by the choice of seeds. From Equation (6) any pair of streams are related by,

$$X_{i,j} = (X_{i,k} + A^i(X_{0,j} - X_{0,k})) \bmod m \tag{10}$$

Therefore, any relationship between the initial seeds will be maintained (mod m) across each ith value. In particular, if the initial seeds are linearly related,

$$X_{0,j} = X_{0,1} + b(j-1) \quad \text{for} j = 1,2,\ldots,n \tag{11}$$

then each set of ith values are also linearly related,

$$X_{i,j} = (X_{i,1} + bA^i(j-1)) \bmod m \tag{12}$$

Similarly, any other simple mathematical relationship between the seeds will be maintained in the ith random numbers.

Figure 1 (Davies and Cooper, 2002) shows an example of the linear patterns that arise, in this case for 65 streams for the 8th value of Lewis's LCG for a particular choice of linear seeds. Although each set of ith values are linearly related, the form of the linear relationship is different for each i. For example, Figure 2 shows the first three values for the same generator but a different choice of seeds.

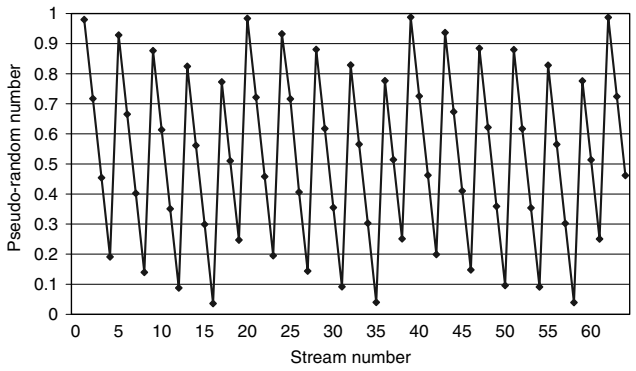

Figure 1 (Davies and Cooper, 2002). The pseudo-random numbers from the 8th number in each stream, where the streams are generated using Lewis's congruential generator and the seed for stream j is $517\,006\,612 + 327\,673\,276j$ (note that there is no significance about the choice of numbers).

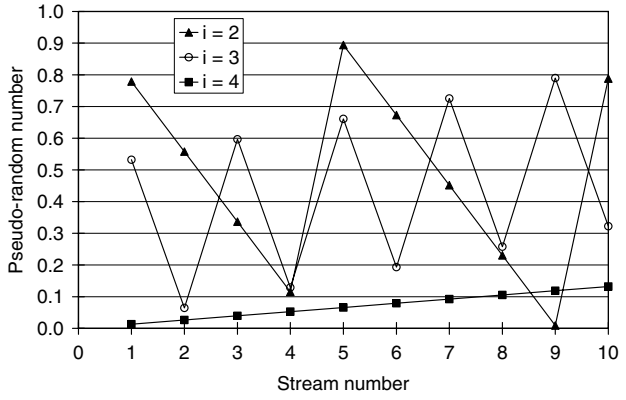

Figure 2 Patterns of the 2nd, 3rd and 4th random numbers ($i = 2$, 3 and 4) of 10 streams from Lewis's congruential generator and seeds = 100 000 × stream number.

5. Examples of problems in application

The question arises as to how important this is. Let us suppose that each of the characteristics, decisions and distributions of the entities are sampled from separate random number streams in this way. If the entities pass though the simulation in an orderly fashion, then the behaviour of the ith entity will be determined by the ith random number of each of the streams where each activity is sampled from a different stream. Hence, all the random numbers used for the ith entity will be linearly related introducing potentially very serious unintended correlations. In simulations where the routes of the entities vary, then this problem may be much less significant.

Problems may also arise in complex simulations. To apply the common random numbers method, entities with identical simulated characteristics should engage in the same activities in the same order between scenarios in each replication. These conditions fail when entities start activities in different orders in the two comparative scenarios and they have different attributes that influence future events. This was overcome in the simulation of screening for *Helicobacter pylori* (Davies *et al*, 2002) by giving each entity a set of random numbers at the beginning of the simulation. However, if each activity time was sampled from a different stream and given to each entity in order, then unexpected results occurred with some people being very much more likely to get several ulcers than others. This was because the ith number in each random number stream was allocated to the same entity and hence (arising from the relationship identified in this paper) the entity had highly correlated times between ulcers. The problem was solved by generating all the pre-allocated numbers from one stream.

6. Solutions

A possible solution is to make a more careful choice of the initial seeds. Law (2007) suggests choosing seeds for congruential generators a large distance apart in the sequence of numbers produced. For a good full period generator with a single cycle, this partitions the cycle into separate sections. For example, they suggest that streams of length 100 000 can be generated using numbers, 100 000 places apart in the sequence, as the seeds. However, this use of equally spaced numbers in the sequence is rejected by both L'Ecuyer (1990) and Kleijnen and Groenendaal (1992) because, even then, strong correlations can occur between the streams. Another possibility suggested by L'Ecuyer (1990) is to use completely different generators for the different streams although he shows that, for an LCG, this requires more than simply changing the value of the constant c. However, this method could be very difficult to apply, particularly when using a software package with a supplied fixed generator. L'Ecuyer (1994) also comments that it may be feasible to obtain the seeds by a truly random process (eg drawing numbered tickets from a hat).

In practice, it would appear that linearly related seeds are often used when applying the common random numbers method and that the issue highlighted in this section is not generally known. In the type of problem described in the previous section, this can invalidate the experiments. In simulation, consideration should be given to both the choice of the RNG and the way in which it is used. In the case of a software package or a supplied random number function, this involves finding out about the algorithm used. For example, the help files for Witness (Lanner Group Ltd., Redditch, UK) state that it uses the combined MRG of L'Ecuyer *et al* (2002). The user of Witness specifies streams rather than seeds, where the streams are partitions of the random number sequence, and so this avoids the issue described here. If there is doubt about the quality of a generator, then the random numbers produced can be tested in various ways (eg L'Ecuyer, 1994; Pidd, 2004; Law, 2007), and this could include plots of multiple streams (as in Figure 2).

7. Conclusion

The use of common random number streams is a popular and effective way of reducing variance. The conventional approach is to allocate different streams to each distribution. This is appropriate where entities performing each activity are identical or else retain their order between scenarios. Where there are activities in which they do not retain their order, it may be necessary to pre-allocate random numbers for these activities to entities when they are created.

We have shown, however, that the use of linearly related seeds, or indeed seeds that are related in any simple mathematical way, from a multiple recursive or congruential RNG leads to the relationship being maintained (subject to modulo arithmetic) between the ith number in each stream for every value of i. This may result in unintended correlations in the behaviour of entities in the simulation and yet this problem does not appear to be generally known or reported in the literature. This problem is particularly serious where entities retain their order between activities, such as in production line models. The use of randomly chosen initial seeds should avoid the type of regular patterns illustrated in the paper, although more research is required on suitable ways of using these generators when implementing the common random numbers method.

Acknowledgements—We are grateful for helpful discussions with Sam Savage, Consulting Professor of Management Science and Engineering, Stanford University and Linus Schrage, Professor Emeritus of Operations Research and Operations Management at the Graduate School of Business of the University of Chicago.

References

Anderson NH and Titterington DM (1993). Cross-correlation between simultaneously generated sequences of pseudo-random uniform deviates. *Statist Comput* **3**: 61–65.

Brooks RJ and Robinson S (2001). *Simulation* with Lewis C (2001). *Inventory Control*, Operational Research Series. Palgrave: Basingstoke.

Davies R and Cooper K (2002). Reducing the computation time of discrete event simulation models of population flows. In: Eldabi T, Robinson S, Taylor SJE and Wilcox PA (eds). *Proceedings of The Operational Research Society Simulation Study Group Workshop*. The Operational Research Society: Birmingham, pp 63–68.

Davies R *et al*. (2002). A simulation to evaluate screening for helicobacter pylori infection in the prevention of peptic ulcers and gastric cancers. *Health Care Manage Sci* **5**: 249–258.

Kleijnen J and Van Groenendaal W (1992). *Simulation a Statistical Perspective*. John Wiley: Chichester.

Law AM (2007). *Simulation Modeling and Analysis*, 4th edn. McGraw-Hill: New York.

L'Ecuyer P (1990). Random numbers for simulation. *Commun ACM* **33**(10): 85–97.

L'Ecuyer P (1994). Uniform random number generation. *Ann Opns Res* **53**: 77–120.

L'Ecuyer P, Simard R, Chen EJ and Kelton WD (2002). An object-oriented random-number package with many long streams and substreams. *Opns Res* **50**(6): 1073–1075.

Lehmer DH (1951). Mathematical methods in large-scale computing units. *Ann Comput Lab Harvard Univ* **26**: 141–146.

Lewis PA, Goodman AS and Miller JM (1969). A pseudo-random number generator for the IBM 360. *IBM Syst J* **8**: 136–146.

Park SK and Miller KW (1988). Random number generators: Good ones are hard to find. *Commun ACM* **31**: 192–1201.

Pidd M (2004). *Computer Simulation in Management Science*, 5th edn. John Wiley: Chichester.

Press WH, Flannery BP, Teukolsky SA and Vetterling WT (1992). *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edn. Cambridge University Press: New York.

Savage S, Schrage L, Lewis P and Empey D (1994). The bad seeds—A parallel random number generation problem weeded out long ago, crops up again, Unpublished manuscript dated 14th January 1994 (presented at the 35th TIMS/ORSA Joint National Meeting in Chicago in 1993).