The Bad Seeds - A Parallel Random Number Generation Problem Weeded Out Long Ago, Crops Up Again

by

Sam Savage
Department of Operations Research
Terman Engineering Center
Stanford University
Stanford, CA 94305

Linus Schrage
Graduate School of Business
1101 E. 58th Street
University of Chicago
Chicago, IL 60637

Peter Lewis
Operations Research
Naval Postgraduate School
Monterey, Ca 93943-5000

David Empey
University of California, Santa Cruz

14 January 1994

Subject Classifications: Simulation: random variable generation, design of experiments

Abstract

A recurrent theme in systems simulation is the need to generate multiple independent sequences from the same random number generator. The advent of parallel computing has accentuated this need. Conventional advice is that if non-overlapping sequences are chosen from a good generator, then the cross-correlation between sequences will be close to zero. When multiple streams of pseudo-random numbers are generated by any of four popularly recommended multiplicative prime modulus generators, we show that there exist initial seeds that avoid the overlap problem but result in dramatically nonrandom behavior. We then give a suggestions for selecting initial seeds that avoid both problems.

History

In 19?? John Von Neumann stated that anyone attempting to generate a random process through deterministic means was living in a state of sin (date and exact quote needed). So far, history has proven him correct on several occasions. Perhaps the most striking of these was the discovery by Marsaglia in 19?? (reference) of a serious flaw in a widely used generator of the time. In an article entitled "Random numbers lie mainly in the plane" he showed that triples of random numbers which should have uniformly filled the unit cube, were in fact distributed accross a number of parallel planes. In 19?? Donald Knuth (ref) described his effort to generate random numbers through an extremely complex iterative process. The output appeared random at first, then quickly converged to a single constant.

The above notwithstanding, it is still tempting to generate pseudo random variables deterministically in order to get repeatable results.

Multiplicative congruential generators

Multiplicative congruential random number generators are probably the most widely used procedures for computer generation of random numbers. Conceptually the technique can be understood in terms of a common clock as follows:

Initialization: With the clock's hands both straight up (12 O'clock), the hour hand is moved to an initial position, x_0 minutes. x_0 is known as the seed.

As the hour hand is moved, the minute hand turns through 12 times as many minutes, perhaps passing 12 O'clock, to arrive at x_1 minutes. Where $x_1 = 12 * x_0 \bmod 60$ is the first random variate.

Iterative step: The hour hand is moved to x_i minutes (the ith random variate), whereupon the minute hand will turn to $x_{i+1} = 12 * x_i \bmod 60$ minutes (the i+1st random variate).

From a practical standpoint, a standard clock is most unsatisfactory at generating pseudo-random numbers, as x_i will cycle through the sequence 12, 24, 48 and 36 indefinitely for any x_0 which is not a multiple of 5, and will remain at 0 for any x_0 which is a multiple of 5. However, by generalizing to a clock with P minutes and a gear ratio of 'a" usefull results may be obtained. For good choices of a and P the generator will be full cycle, that is, the generator will enumerate every integer in the interval [1, P 1] exactly once before it cycles. Thus, from a global perspective there is no reason to prefer one seed over another. Furthermore the sequence $\{x_i\}$ will display no serial correlation.

Now suppose we wish to generate two sequences of numbers {xi} and {yi} and we wish to use the same recursion for both, i.e., xi = f(xi 1) and yi = f(yi 1). If we are concerned about the correlation between xi and yi, can we still say that from a global perspective we are indifferent to choices of the initial seeds x0 and y0? For standard multiplicative congruential generators the answer is "no".

A number of simulation references suggest implicitly that when generating multiple streams using the same generator, the starting seeds should be far apart, see for example, L´Écuyer and Côté (1991) or Bratley, Fox, and Schrage (1987). The GPSS/H system, for example, uses starting seeds that are 100,000 apart, see Schriber (1991). Below we show that for standard generators, "seeds far apart" is not sufficient, specifically that there is a second potential problem independent of overlap.

A standard multiplicative congruential generator uses the recursion xi = a*xi 1 mod P where a is the multiplier and P is the modulus. If P is a prime number bigger than 2 and a is a primitive root modulo P, then the generator is full cycle, i.e., if 0 <x0 <P, then xi will take on every value in [1, P 1] exactly once before a number is repeated. If P = 2^31 1, then approximately 25% of the integers in [2, P 1] are primitive roots. This full cycle behavior is desirable but does not guarantee that the x_i's have good statistical qualities (such as lack of autocorrelation). Fishman and Moore (1986) point out that among multipliers that are full cycle, some multipliers are much better than others in this statistical regard.

Suppose we choose two integer seeds $0 < x_0, y_0 < P$. Now there must exist some integer b such that $y_0 = b*x_0 \bmod P$. For example, if y_o and x_o are k apart in the full stream, then $b = a^k \bmod P$ works. For subsequent "draws" we can write:

$$\begin{aligned} y_i &= a*y_{i\ 1} \bmod P \\ &= a^i*y_0 \bmod P \\ &= a^i*b*x_0 \bmod P \\ &= b*a^i x_0 \bmod P \\ &= b*x_i \bmod P \end{aligned}$$

Thus, we can see x_i and y_i are effectively two successive numbers generated by a generator of the form:

$$z_{i+1} = b*z_i \bmod P.$$

From a global perspective (x_i, y_i) have exactly the same statistical behavior as the pair (z_i, z_{i+1}). Thus, if b is a good multiplier in the statistical sense of Fishman and Moore, then we would expect x_i and y_i to be independent and therefore uncorrellated. Greenberger (1961) shows that the serial correlation of (z_i, z_{i+1}) is bounded from above by

$$(1/b) + (b+6)/P.$$

If b is not much bigger than 1 or not much smaller than P, this is not a very reassuring bound. The following example shows that this concern is justified.

Example

Suppose we carelessly choose $x_0 = 100{,}000$, and $y_0 = 200{,}000$. In this case it is easy to see that b = 2. Whenever $x_i < .25*P$, then $y_i < .5*P$ and so we would expect that x_i and y_i would appear dependent.

For a generator of the form $x_i = ax_{\ i\ 1}$ Mod P, Figure 1 shows the relationship between x_i and y_i for the case of b = 2. For arbitrary integer b, one needs at most b parallel lines to cover the pairs (x_i, y_i).

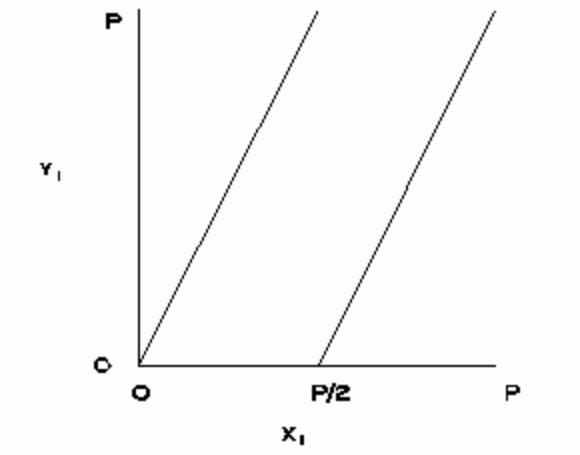

Figure 1. x_i vs. y_i, when $y_o = 2*x_o$

Choosing the tuple multiplier, b, equal to 2, implies that for 0 x_i P/2, we have $y_i = 2*x_i$; while for P/2 < x_i < P, we have $y_i = 2*x_i$ P. If we look at $S_i = x_i + y_i$, then with probability .5, S_i is uniform on (0, 1.5*P) and with probability .5, S_i is uniform on (.5*P, 2*P). Thus, S_i has the "Olympic Awards Pedestal" distribution as in Figure 2; decidedly different from triangular distribution we would expect if x_i and y_i were independent.

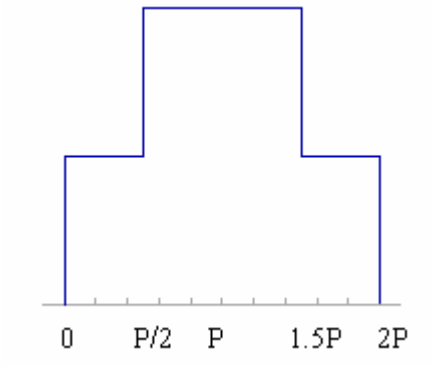

Figure 2: Distribution of $x_i + y_i$

Thus, b = 2 or 3 is probably a bad choice, even though 2 and 3 occur at steps 1,385,473,320 and 1,783,741,719 respectively after 1 in the sequence generated by the popularly recommended multiplier 16807, see Lewis et. al. (1969).

We have illustrated that choosing b a small integer is not good. This result can be generalized slightly to small rational numbers. Suppose y_o and x_o are chosen so that there are positive integers b and c such that

$c\ y_o \bmod P = b\ x_o \bmod P$

We then claim that for i = 1, 2, ...:

$c\ y_i \bmod P = b x_i \bmod P$.

Suppose it is true for i 1; multiply by a gives:

$a\ c\ y_{i\ 1} \bmod P = a\ b\ x_{i\ 1} \bmod P$

or $c\ a\ y_{i\ 1} \bmod P = b\ a\ x_{i\ 1} \bmod P$

or $c\ y_i \bmod P = b\ x_i \bmod P$, as was claimed.

We now claim that the (x_i, y_i) fall on at most b+c 1 parallel lines. (They may in fact fall on fewer than b+c 1 lines, - which is even worse). The argument is as follows:

$c\ y_i \bmod P \quad bx_i \bmod P = 0$ implies

that there is an integer k, possibly negative, such that

$c\ y_i \quad bx_i \quad kP = 0$.

Because $0 < x_i, y_i < P$, the largest k needed is such that: c(P 1) b kP = 0. So k (cP (c+b))/P < c. Similary, the smallest k needed is such that:

c b (P 1) kP = 0

so

k (bP + c+b)/P > b

The number of integers strictly between +c and b is c+b 1. Each value of k corresponds to a line in (x,y) space, so there are at most c+b 1 such lines.

De Matteis and Pagnutti (1988) analyze the case $b = a^{(P\ 1)/2} \bmod P$, c = 1 where P is a prime and a is a primitive root of P. This choice of b is equivalent to jumping ahead halfway through the full cycle, that is, $y_i = x_{i+s}$ where s = (P 1)/2. They show the unhappy result that in this case $y_i = P\ x_i$, that is, perfect negative correlation. They go on to say: "We have also analyzed the pairs (x_i, x_{i+s}) for values of s corresponding to other divisors...but we did not succeed in detecting strong linear patterns." We show that there are seeds that do display strong linear patterns.

Combinations of Generators

Wichman and Hill (1982) describe a generator obtained by combining the outputs of three generators, while L´Écuyer (1988) describes two generators obtained by combining the outputs of either two or three generators. The same weakness that we have just described for simple generators afflicts these more complicated generators. Specifically, an inappropriate choice of seeds for two parallel streams X and Y can produce correlation that is essentially identical to that observed in Figure 1.

The "two stream" combination generator described in L´Écuyer (1988) is of the form:

$X1(i) = a_1 X1(i\ 1) \bmod P_1$,
$X2(i) = a_2 X2(i\ 1) \bmod P_2$,
X3(i) = X1(i) X2(i)
If X3(i) 0, then X3(i) = X3(i) + (P_1 1)

The output is the stream X3(i). The user sees the component streams X1(i) and X2(i) only to the extent that he must supply the seeds X1(0) and X2(0).

L´Écuyer's recommends:

a_1 = 40014, P_1 = 2147483563, a_2 = 40692, and P_2 = 2147483399.

Now suppose the user needs a second stream generated by the same generator but with different seeds. Replace X by Y to denote this stream. The user happens to choose seeds such that $Y1(0) = b*X1(0) \bmod P_1$ and $Y2(0) = b*X2(0) \bmod P_2$. Suppose a_1, a_2, P_1 and P2 are such that the X1(i) and X2(i) streams are full cycle, and $w = P_1 \quad P_2 > 0$. The qualitative result is that if both b and w are small relative to P_1, Y3(i) is well approximated by $bX3(i) \bmod P_1$. In fact if b = 2 and Y3(i) is plotted vs. X3(i), the graph for L´Écuyer's generator is indistinguishable from Figure 1.

Suppose we randomly choose the i^{th} output from the Y stream. By previous arguments we have that $Y1(i) = bX1(i) \bmod P_1$. Now $Y2(i) = bX2(i) \bmod P_2 = bX2(i) \quad rP_2 = bX2(i) \quad r(P_1\ w) = bX2(i)\ rP_1 + rw$

where $r = \lfloor bX2(i)/P_2 \rfloor$,
and $\lfloor\ \rfloor$ denotes the integer part.

so

$Y2(i) \quad rw = bX2(i) \quad rP_1$.

If Y2(i) rw 0, then it follows that

$Y2(i) \quad rw = bX2(i) \mod P_1$.

Because i was chosen randomly, Y2(i) is uniform distributed over [1, P_2 1]. Thus, Prob{Y2(i) rw} = 1 rw/(P_2 1) for rw P_2 1. Because $X2(i) < P_2$, it follows that r < b, so Prob{Y2(i) rw} > 1 bw(P_2 1).

So for $bw << P_2$, with high probability:

$Y3(i) = b\ X1(i) \bmod P_1 \quad \{b\ X2(i) \bmod P_1 + rw\} + (P_1 \quad 1)$

where = 1 if Y3(i) would otherwise be 0.

We can rewrite this as

$Y3(i) = b\ \{X1(i) \quad X2(i)\} \quad rw \quad r_1\ P_1$

where r_1 is the unique integer causing:

$0 < b\{X1(i) \quad X2(i)\} \quad rw \quad r_1 P_1 < P_1$

Now consider $bX3(i) \bmod P_1$

$= b[X1(i) \quad X2(i) + \quad _1 (P_1 \quad 1)] \bmod P_1$

where $_1 = 1$ if $X1(i) \quad X2(i) \quad 0$, else 0;

Rearranging:

$bX3(i) \bmod P_1 = b\{X1(i) \quad X2(i)\} \quad b_1 \quad r_2 P_1$

where r_2 is the unique integer causing:

$0 < b\{X1(i) \quad X2(i)\} \quad b_1 \quad r_2 P_1 < P_1$

If P_1 and P_2 are both prime, and a_1 and a_2 are full cycle multipliers, then every possible value of X1(i) is equally likely, given the value of X2(i). We can argue that given the value of X2(i), the fraction of the values of X1(i) that cause $r_2 \quad r_1$ is less than $2\ b(w+2)/P_1$. Recalling that $0 \quad , \quad 1$ 1 we conclude that

$\text{Prob}\{| Y3(i) \quad bX3(i) \text{ mode } P_1 | < b + bw + 1\}$

$[1 \quad bw/(P_2 \quad 1)]\ [1 \quad 2b\ (w + 2)/P_1]$

For L´Écuyer's two stream combination generator, w = 2147483563 2147483399 = 164. If we carelessly choose seeds so that b = 2, then for the median value of Y3(i) with probability about .9999995 we will have that $|(Y3(i) \quad bX3(i) \bmod P_1)/Y3(i)| \quad .0000003$.

We do not give proofs here for the other generator in L´Écuyer (1988) or Wichman and Hill (1982), however, the same effect illustrated in Figure 1 can be observed by choosing seeds in the second generator that are a small multiple of the seeds in the first generator.

Choosing Multiple Seeds Systematically

Thus, we have several negative results on how not to choose other seeds. Can we give any positive recommendations?

Suppose we wish to generate m parallel random sequences from a simple generator. Think of these as m-tuples $(x_{i1}, x_{i2}, \ldots x_{im})$, $(x_{i+1,1}, x_{i+1,2}, \ldots, x_{i+1,m})$, etc., where x_{ij} is the ith number from the jth sequence. Given our previous comments, a way of doing this is to use two different simple generators. Generate $x_{i1}, x_{i+1,1}, \ldots$ by the recursion: I) $x_{i+1,1} = a*x_{i,1} \bmod P$, and generate x_{ij}, for j = 2, 3, ... m, by the recursion: II) $x_{ij} = b*x_{ij\ 1} \bmod P_1$. From our earlier result we know we can achieve the same effect using recursion (II) when i = 1 and thereafter replacing (II) by the recursion: II) $x_{ij} = a * x_{i\ 1,j} \bmod P$ for j = 2, 3, ..., m. Conceptually, the above approach randomly draws m-tuples of successive numbers from the sequence of numbers generated by $z_{i+1} = b*z_i \bmod P_1$. Where in this sequence we make the draw is determined by the generator $x_{i+1,1} = a*x_{i,1} \bmod P$.

What are the advantages of using this approach? There are three attractions: i) for a given i, the tuple $(x_{i1}, x_{i2}, \ldots x_{im})$ has all the statistical properties associated with the multiplier b, ii) for a given j, the tuple $(x\ x_{i+1,j}, \ldots x_{i+k,j})$ has all the statistical properties associated with the multiplier a, and iii) the time to generate a (pseudo) random deviate is that required by the generator using the multiplier a. In particular, the generator using any of the multipliers a = 16807, 39373, 48271, or 69621 and P = 2^31 1 is fast and particularly easy to implement in a portable fashion, see Park and Miller (1988) and Bratley, Fox, and Schrage (1987).

If we use this approach, how should we choose b to ensure both satisfactory statistical and overlap properties? Fishman and Moore (1986) implicitly examined the statistical properties of every possible full cycle multiplier for generators of the form $z_{i+1} = b*z_i \bmod (2^{31} 1)$. Based on number theoretic and empirical tests they recommend the following five multipliers that appear in the left most column of the Table 1.

Table 1. Suggested Seeds.

b_1	b_2	b_3	b_4	b_5	Minimum Separation
742938285	1710921057	1796558312	1943891214	1800077045	159,644,913
950706376	129027171	1728259899	365181143	1966843080	171,352,421
1226874159	604629562	407791863	679979433	557612409	268,384,731
62089911	847344462	1061653656	1954074819	226824280	341,462,557
1343714438	389745688	252992993	1742312917	988214982	13,274,813

Note: $b_j = bj_1 \bmod (2^{31} 1)$ for each row, for j = 1 to 5.

Using any one of the rows in Table 1 and the approach just described, one can generate starting seeds for up to six streams. Seed x_{01} may be chosen arbitrarily, then seed x_{0j} is obtained by $x_{0j} = b_j * x_{01} \bmod (2^{31} 1)$, for j = 1 to 5.

Regarding overlap, the last column in the table shows minimum separation between any two seeds of the row in the sequence produced by the generator $x_{i+1} = 16807 * x_i \bmod (2^{31} 1)$. This shows that the first four sets of b values result in 6-tuples that can be run for over 100,000,000 iterations without fear of overlap.

Effectively, each row of the table is the first six outputs of one of the generators recommended by Fishman and Moore, initiated with a seed of 1. An arbitrary number of parallel streams can be initiated by making additional draws from these generators. Obviously, as more streams in parallel are required, the possibility of getting nonoverlapping sequences diminishes. Portable versions of the five generators are available from the second author. They require at most two to three times the computation per iterate of the "fast" generator with a = 16807.

Summary

We have seen that for widely used and tested generators based on the multiplicative congruential generator, choosing starting seeds carelessly can lead to very nonrandom results.

References:

Bratley, P., B. Fox and L. Schrage (1987), A Guide to Simulation, 2nd ed. Springer-Verlag, New York.

De Matteis, A. and S. Pagnutti (1988), "Parallelization of Random Number Generators and Long-Range Correlations", Numerische Mathematik, vol. 53, pp. 595-608.

Fishman, G.S. and L.R. Moore (1986), "An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus 2^{31} 1. SIAM J. Sci. Statistical Computing, vol. 7, pp 24-45.

Greenberger, M. (1961), "Notes on a New Pseudo-random Number Generator", J. of ACM, vol. 8, pp. 163-167.

L´Écuyer, P. (1988), "Efficient and Portable Combined Random Number Generators", Comm. of ACM, vol. 31, no. 6, pp. 742-774.

L´Écuyer, P. and S. Côté (1991), "Implementing a Random Number Package with Splitting Facilities", ACM Transactions on Mathematical Software, vol. 17, no. 1, pp. 98-111.

Lewis, P.A.W., Goodman, A.S., and Miller, J.M. (1969). A Pseudo-random Number Generator for the System/360. IBM Systems Journal, 8, 136-146.

Park, S.K., and K.W. Miller (1989) "Random Number Generators: Good Ones are Hard to Find," Comm. of ACM, vol. 31, no. 10, pp. 1192-1201.

Schriber, T. (1991), "An Introduction to Simulation Using GPSS/H", John Wiley and Sons, New York.

Wichman, B.A. and I.D. Hill (1982), Appl. Stat., vol. 31, pp. 188-190.