SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NY 14853-3801

TECHNICAL REPORT NO. 1055

May 1993

On the Problem of Overlapping Pseudo-Random Number Streams: A Queuing Simulation Example[1]

by

F. Chance

[1]Research supported by the Defense Advanced Research Projects Agency.

ABSTRACT

Several popular implementations of linear congruential algorithms allow the generation of pseudo-random numbers from separate streams. Streams are actually different starting points in the total cycle traveled by the generator. Thus it is possible to generate enough variates from any given stream to overlap the variates generated from other streams. We give an example, a simulation of an M/M/1 queue, where the natural application of multiple streams can lead to erroneous estimates of mean queue delay. For a simple system where interarrival and service times are uniform, we obtain analytic expressions for the queue delay, and conclude that the presence of overlapping streams violates the assumption of independence between service and interarrival times.

1 INTRODUCTION

When modeling queuing systems, independence between the arrival and service time processes is often assumed. Thus, when simulating these models, it is natural to draw the pseudo-random interarrival and service times from separate, "independent" streams, if provided by the generator at hand.

However, in the course of the author's research, a close variant of the program in Section 2 was constructed, and found not to agree with analytic expressions for the queue delay in an M/M/1 queue. The cause has been traced to the use of streams, where enough pseudo-random variates have been generated so as to make the streams overlap.

On a simplified model, it is possible to obtain some analytic results for the expected queue delay, when overlapping streams are present, and it becomes clear that this overlapping behavior violates the independence assumption between the arrival and service time processes. This note is intended as a cautionary tale on the troubles that can arise when overlapping streams are ignored.

In Section 2, we give an example where multiple streams leads to erroneous estimates of queue delay in an M/M/1 queuing simulation. In Section 3, we briefly outline the operation of linear congruential pseudo-random number generators. In Section 4, we present an extreme case where analytic results are obtainable, showing the impact of multiple streams versus a single

stream.

2 EXAMPLE: THE M/M/1 QUEUE

In this section we give a queuing simulation example where overlapping pseudo-random numbers results in an erroneous estimate of mean queue delay. We simulate a single-server queue, with a first-come-first-served discipline.

The times between customer arrivals are modeled as independent, identically distributed (i.i.d.) exponential random variables, with mean $1/\lambda$. Similarly, service times are modeled as i.i.d. exponential random variables with mean $1/\mu$. Service times are assumed independent of interarrival times. In this example, $\lambda = 1.0$, and $\mu = 1.10$, giving a traffic intensity of $\lambda/\mu \approx 0.91$.

We use a C version of the linear congruential generator given in Law and Kelton (1991). To simulate successive waiting times, we use the Lindley recursion (Lindley 1952),

$$W_{n+1} = (W_n + S_n - A_n)^+, \quad n \geq 0,$$

where W_n is the time spent in queue waiting for service by customer n, S_n is the service time of customer n, and A_n is the time between the arrivals of customers n and $n+1$.

Suppose we wish to obtain five thousand replications of the customer waiting times for customers one through one hundred. One possible way of programming this simulation is shown in Figure 1. The pseudo-random number generator is shown in Figure 2.

If we compile this program with the symbol "MULTIPLE_STREAMS" defined, interarrival times are generated from stream one, service times from stream two. If the symbol "MULTIPLE_STREAMS" is not defined, all pseudo-random variates are generated from stream one.

Figure 3 shows a comparison between the analytic value for queuing delay, the results from the multiple streams run, and the results from the single stream run. The analytic values for queuing delay were calculated using the program described in Kelton and Law (1985). Clearly, the results obtained when multiple streams are used does not agree with either the analytic or the single stream results.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define MAX(a,b) ( (a) > (b) ? (a) : (b) )
main()
{
  double *W,S,A,sum,sum_sq,lambda,mu,avg,var,krand();
  int n_reps,s_stream,a_stream,rep,n;

  n_reps = 5000; lambda = 1.0; mu = 1.1;
  if ((W = (double *) calloc(n_reps+1,sizeof(double)))
      == NULL) {
    printf("Unable to get memory.\n");
    exit(1);
  }
#ifdef MULTIPLE_STREAMS
  a_stream = 1; s_stream = 2;
#else
  a_stream = 1; s_stream = 1;
#endif
  for(n=1; n<=100; n++) {
    sum = 0.0; sum_sq = 0.0;
    for(rep=1; rep<=n_reps; rep++) {
      A = -log(krand(a_stream))/lambda;
      S = -log(krand(s_stream))/mu;
      W[rep] = MAX(0,W[rep]+S-A);
      sum += W[rep]; sum_sq += W[rep]*W[rep];
    }
    avg = sum/n_reps; var = (sum_sq-n_reps*avg*avg)/(n_reps-1);
    printf("Customer %3d: Avg delay: %6.3f, Std Err: %6.3f\n",
   n,avg,pow(var/n_reps,0.5));
  }
}
```

Figure 1: Main program for simulating an M/M/1 queue.

```
/* Prime modulus multiplicative linear congruential generator
   Z[i] = (630360016 * Z[i-1]) (mod(pow(2,31) - 1)), based on
   Marse and Roberts' UNIRAN.  C version of the FORTRAN in
   Law & Kelton 1991 ''Simulation Modeling and Analysis''. */

long zrng[] = {0, 1973272912, 281629770};

#define MODLUS 2147483647
#define MULT1       24112
#define MULT2       26143

/* Generate the next random number. */

double krand(stream)
int stream;
{
    static long zi, lowprd, hi31;

    zi     = zrng[stream];
    lowprd = (zi & 65535) * MULT1;
    hi31   = (zi >> 16) * MULT1 + (lowprd >> 16);
    zi     = ((lowprd & 65535) - MODLUS) +
             ((hi31 & 32767) << 16) + (hi31 >> 15);
    if (zi < 0) zi += MODLUS;
    lowprd = (zi & 65535) * MULT2;
    hi31   = (zi >> 16) * MULT2 + (lowprd >> 16);
    zi     = ((lowprd & 65535) - MODLUS) +
             ((hi31 & 32767) << 16) + (hi31 >> 15);
    if (zi < 0) zi += MODLUS;
    zrng[stream] = zi;
    return ((zi >> 7 | 1) + 1)/ 16777216.0;
}
```

Figure 2: Code for pseudo-random number generator.

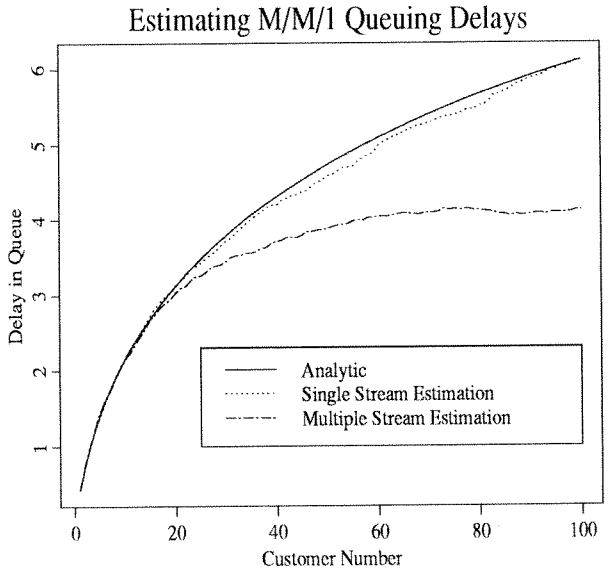

Figure 3: A comparison between analytic expressions and simulation results for queuing delay of the first one hundred customers in an M/M/1 queue with arrival rate $\lambda = 1.0$ and service rate $\mu = 1.10$.

3 LINEAR CONGRUENTIAL GENERATORS

In this section, we briefly describe the linear congruential class of pseudo-random number generators, and the way in which streams are implemented.

For a complete description, and for other classes of generators, see Bratley, Fox, and Schrage (1987). In general, the nth integer generated is a function of the previous number in the sequence,

$$X_n \quad = \quad (aX_{n-1} + c) \bmod m, \quad n \geq 1.$$

The generator we use has $c = 0$, and is called a "multiplicative congruential" generator. If a number in the range $(0, 1)$ is desired, then X_n is divided by m before being returned.

Various choices of a and m have been argued in the literature, based on theoretical and empirical considerations. For specific recommendations see Bratley, Fox, and Schrage (1987) and references therein.

If certain constraints on a and m hold, the generator is full cycle. That is, for any value of X_0, all other integers in $[1, m-1]$ will be generated before X_0 is repeated. The "seed" for this generator is the value X_0.

Streams can be implemented for this type of generator by obtaining a list of seeds that are a guaranteed number of iterations apart. For example, given any value of X_0, we could use X_0, $X_{100,000}$, $X_{200,000}$ and so on as the seeds for our streams, with the result that streams contain one hundred thousand non-overlapping numbers. This approach is the one taken in the generator we use in our example.

One side effect of this approach is that if more than one hundred thousand numbers are generated from a stream, the numbers generated overlap with those generated from subsequent streams. While it may seem the results of this effect should be negligible, the example given in Section 2 shows that the outcome can be far from harmless.

4 ANALYTIC RESULTS: THE $U/U/1$ QUEUE

In order to understand why overlapping pseudo-random number streams can cause the problems described in Section 2, this section presents some analytic results for expected waiting times in the simulation of a simplified single-server queue.

Suppose we have a single-server queue, where the interarrival times are i.i.d. uniform$(0, 1)$ variates, and service times are also i.i.d. uniform$(0, 1)$. Although this system has a traffic intensity of one, it is reasonable to estimate the expected value of the waiting time for customer n, for any finite n. We make the assumption, although it will be false for the case of multiple streams, that service times are independent of interarrival times.

We use the program of Figures 1 and 2, modified to produce uniform interarrival and service times, with the number of replications set to one hundred thousand for simplicity.

Denote by U_n^s the nth pseudo-random number generated from stream s, $n \geq 1$. If the streams are adjacent in the generator's cycle,numbered consecutively from one, and contain one hundred thousand iterations, then

for $s > 1$,

$$U_n^s = U^1_{(s-1)*100{,}000+n}. \tag{1}$$

Denote by W_n^j the waiting time for customer n in replication j. Similarly, denote by A_n^j the interarrival time between customers n and $n+1$ in replication j. Finally, denote by S_n^j the service time of customer n in replication j.

Waiting times for the first several customers are given below,

$$\begin{aligned} W_0^j &= 0.0, \\ W_1^j &= (S_0^j - A_0^j)^+, \\ W_2^j &= (W_1^j + S_1^j - A_1^j)^+. \end{aligned}$$

We compute the expected value of W_2^j, and note the difference caused by using multiple streams versus a single stream.

First we examine the case of multiple streams. Under this operating procedure,

$$\begin{aligned} A_0^j &= U_j^1, \\ S_0^j &= U_j^2, \\ A_1^j &= U^1_{100{,}000+j}, \\ S_1^j &= U^2_{100{,}000+j}. \end{aligned}$$

But according to (1),

$$\begin{aligned} S_0^j &= U_j^2 \\ &= U^1_{100{,}000+j} \\ &= A_1^j, \end{aligned}$$

and clearly the service times and interarrival times are dependent. Hence, W_2^j becomes

$$\begin{aligned} W_2^j &= (W_1^j + S_1^j - A_1^j)^+ \\ &= ((S_0^j - A_0^j)^+ + S_1^j - A_1^j)^+ \\ &= ((A_1^j - A_0^j)^+ + S_1^j - A_1^j)^+ \\ &= ((V - W)^+ + Z - V)^+, \end{aligned}$$

where V, W and Z are i.i.d. uniform$(0,1)$ variates. To compute the expected value of W_2^j, we calculate

$$\mathrm{E}[W_2^j] = \int_{z=0}^{1}\int_{w=0}^{1}\int_{v=0}^{1}((v-w)^+ + z - v)^+ dvdwdz. \tag{2}$$

Note that for $v > w$, the integrand becomes

$$(v - w + z - v)^+ = (z - w)^+,$$

while for $v \leq w$, the integrand becomes

$$(z - v)^+.$$

Hence we can write (2) as

$$\begin{aligned}\mathrm{E}[W_2^j] &= \int_{z=0}^{1}\int_{w=0}^{z}\int_{v=w}^{1}(z-w)^+ dvdwdz \\ &+ \int_{z=0}^{1}\int_{v=0}^{z}\int_{w=v}^{1}(z-v)^+ dwdvdz. \end{aligned}\tag{3}$$

Using the algebraic manipulator MACSYMA, (3) evaluates to $1/4 = 0.25$, which is also confirmed by experimentation.

Under the case of a single stream, we can write W_2^j as

$$W_2^j = ((V - W)^+ + Z - Y)^+,$$

where V, W, Z and Y are i.i.d. uniform$(0,1)$.

Proceeding as above, we evaluate

$$\mathrm{E}[W_2^j] = \int_{z=0}^{1}\int_{y=0}^{1}\int_{w=0}^{1}\int_{v=0}^{1}((v-w)^+ + z - y)^+ dvdwdydz. \tag{4}$$

If $v \leq w$, the integrand becomes

$$(z - y)^+.$$

For $v > w$, we must distinguish two cases, $y \leq z$ and $y > z$. When $v > w$ and $y \leq z$, the integrand becomes

$$(v - w + z - y).$$

When $v > w$ and $y > z$, we must have the additional constraint $1 \geq v > w + y - z$ for the integrand to be non-zero. But $w + y - z < 1$ is equivalent to $w < 1 - y + z < 1$, since $y > z$. Hence (4) becomes

$$\begin{aligned} \mathrm{E}[W_2^j] &= \int_{z=0}^{1}\int_{y=0}^{z}\int_{w=0}^{1}\int_{v=0}^{w}(z-y)dvdwdydz \\ &+ \int_{z=0}^{1}\int_{y=0}^{z}\int_{w=0}^{1}\int_{v=w}^{1}(v-w+z-y)dvdwdydz \\ &+ \int_{z=0}^{1}\int_{y=z}^{1}\int_{w=0}^{1+z-y}\int_{v=w+y-z}^{1}(v-w+z-y)dvdwdydz. \end{aligned} \quad (5)$$

Using MACSYMA, (5) evaluates to $17/60 \approx .28333$, again confirmed by simulation. Hence, the expected values of customer waiting times depend on whether or not multiple overlapping streams are used. We see here how overlapping streams can violate the independence assumption between interarrival times and service times.

5 SUMMARY

Many implementations of linear congruential pseudo-random number generators allow the generation of numbers from a variety of different streams. These streams are generally different starting seeds for the generator, spaced an even number of iterations apart. Hence it is possible to generate enough variates from a stream to cross over into those generated from subsequent streams. In queuing simulations, it is natural to use one stream for interarrival times, and another for service times. If replications of the experiment are performed, it is quite easy to generate enough pseudo-random numbers from a stream to overlap those generated from subsequent streams. We give an example, a simulation of an M/M/1 queue, where this overlapping behavior can result in quite erroneous estimates of queuing delays.

To gain intuition about this behavior, we study an even simpler situation, and derive analytic expressions showing the difference between the queue delay for multiple streams versus a single stream. This difference is due to the violation of the independence assumption between the interarrival and service time processes.

We do not seek to impugn the use of streams as a convenient simulation tool. Rather, we wish to clarify the impact that overlapping pseudo-random

number streams can have on simulation estimates, and stress the care that must be taken so this behavior does not violate assumptions about the system being modeled.

ACKNOWLEDGMENTS

The author would like to thank David Goldsman for his assistance with several mathematical difficulties encountered during this work.

REFERENCES

P. Bratley, B. Fox, and L. Schrage. 1987. *A Guide to Simulation.* New York: Springer-Verlag.

W. D. Kelton and A. M. Law. 1985. The Transient Behavior of the M/M/s Queue, with Implications for Steady-State Simulation. *Operations Research* **33**: 378-396.

A. M. Law and W. D. Kelton. 1991. *Simulation Modeling and Analysis.* New York: McGraw-Hill.

D. V. Lindley. 1952. The Theory of Queues with a Single Server. *Proceedings of the Cambridge Philosophical Society* **48**: 277-289.