Generating Uniform Random Numbers

Christos Alexopoulos and Dave Goldsman

Georgia Tech

4th May, 2020

Outline

Introduction

Uniform(0,1) random numbers are the key to random variate generation in simulation.

Goal: Give an algorithm that produces a sequence of *pseudo-random numbers (PRN's)* $U_1, U_2, \ldots$ that "appear" to be iid Unif(0,1).

Desired properties of algorithm

- output appears to be iid Unif(0,1)
- very fast
- ability to reproduce any sequence it generates

References: Law (2015).

Classes of Unif(0,1) Generators

- output of random device
- table of random numbers
- midsquare (not very useful)
- linear congruential (most commonly used in practice)
- Tausworthe (linear recursion mod 2)
- hybrid

Some Generators We Won't Use

a. Random Devices

Nice randomness properties. However, Unif(0,1) sequence storage difficult, so it's tough to repeat experiment.

Examples:

- flip a coin
- particle count by Geiger counter
- least significant digits of atomic clock

b. Random Number Tables

List of digits supplied in tables.
Cumbersome and slow — not very useful.
Once tabled no longer random.

c. Mid-Square Method (J. von Neumann)

Idea: Take the middle part of the square of the previous random number. John von Neumann was a brilliant and fun-loving guy, but his method proved to be lousy!

Example: Take $U_i = X_i/10000$, $\forall i$, where the X_i's are positive integers < 10000.

Set *seed* $X_0 = 6632$; then $6632^2 \rightarrow 43\mathbf{9834}24$;
So $X_1 = 9834$; then $9834^2 \rightarrow 96\mathbf{7075}56$;
So $X_2 = 7075$, etc,...

Unfortunately, positive serial correlation in U_i's.

Also, occasionally degenerates; e.g., consider $X_i = 0003$.

Outline

Linear Congruential Generators

LCG's are the most widely used generators. These are pretty good when implemented properly.

$X_i = (aX_{i-1} + c)$ mod m, where X_0 is the seed.

$U_i = X_i/m,\ i = 1, 2, \ldots$

Choose a, c, m carefully to get good statistical quality and long *period* or *cycle length*, i.e., time until LCG starts to repeat itself.

If $c = 0$, LCG is called a *multiplicative* generator.

Trivial Example: For purposes of illustration, consider the LCG

$$X_i = (5X_{i-1} + 3) \bmod 8$$

If $X_0 = 0$, we have $X_1 = (5X_0 + 3) \bmod 8 = 3$; continuing,

i	0	1	2	3	4	5	6	7	8	9
X_i	0	3	2	5	4	7	6	1	0	3
U_i	0	$\frac{3}{8}$	$\frac{2}{8}$	$\frac{5}{8}$	$\frac{4}{8}$	$\frac{7}{8}$	$\frac{6}{8}$	$\frac{1}{8}$	0	$\frac{3}{8}$

so that the sequence starts repeating with $X_8 = 0$.

This is a *full-period generator*, since it has cycle length $m = 8$. Generally speaking, full-period is a good thing. □

Better Example: Here's a classical LCG (Bratley, Fox, and Schrage 1987):

$$X_i = 16807\, X_{i-1} \bmod (2^{31} - 1).$$

It works well for small applications, is fast, and is full-period with cycle length $>$ 2 billion.

(`ix` is an integer between 1 and $2^{31} - 1$; `k` is also integer.)

Correct implementation:

```
function unif(ix)
k = ix/127773      (integer division leaves no remainder)
ix = 16807*(ix - k*127773) - k*2836
if (ix < 0) ix = ix + 2147483647
unif = ix*4.656612875E-10
return unif
```

So what can go wrong with LCG's?

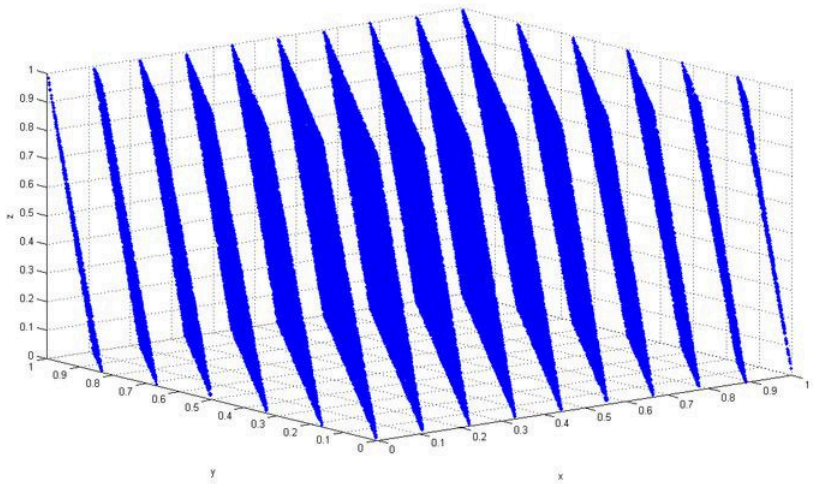
a. Something like $X_i = (4X_{i-1} + 2) \bmod 8$ is *not* full-period, since it only produces even integers.
b. Something like $X_i = (X_{i-1} + 1) \bmod 8$ is full-period, but it produces very non-random output: $X_1 = 1$, $X_2 = 2$, $X_3 = 3$, etc.
c. In any case, if m is small, you'll get quick cycling whether or not the generator is full period. "Small" could mean anything less than 2 billion or so!
d. And just because m is big, you still have to be careful. In addition to a. and b. above, some subtle problems can arise. Take a look at RANDU. . . .

Example: The infamous RANDU generator,

$$X_i \;=\; 65539\, X_{i-1} \bmod 2^{31},$$

was popular during the 1960's.

Here's what the consecutive triplets (U_i, U_{i+1}, U_{i+2}) look like if you plot them in 3-D (stolen from Wikipedia). If they were truly iid Unif(0,1), you'd see dots randomly dispersed in the unit cube. But instead, the random numbers fall entirely on 15 hyperplanes (not good).



Outline

Generalizations of LCGs

A Simple Generalization:

$X_i = \left(\sum_{j=1}^{q} a_i X_{i-j} \right) \bmod m$, where the a_i's are constants.

Extremely large periods possible (up to $m^q - 1$ if parameters are chosen properly).

The Combined Generator MRG32k3a of L'Ecuyer (1999)

- This generator combines two full-period sequences $\{X_{1,i} : i \geq 1\}$ and $\{X_{2,i} : i \geq 1\}$.
- Initialize the seeds $\{X_{1,0}, X_{1,1}, X_{1,2}\}$ and $\{X_{2,0}, X_{2,1}, X_{2,2}\}$.
- For $i \geq 3$, compute

$$\begin{aligned} X_{1,i} &= (1{,}403{,}580\, X_{1,i-2} - 810{,}728\, X_{1,i-3}) \bmod (2^{32} - 209) \\ X_{2,i} &= (527{,}612\, X_{2,i-1} - 1{,}370{,}589\, X_{2,i-3}) \bmod (2^{32} - 22{,}853) \end{aligned}$$

and set

$$\begin{aligned} Y_i &= (X_{1,i} - X_{2,i}) \bmod (2^{32} - 209) \\ U_i &= \begin{cases} Y_i/(2^{32} - 208) & \text{if } Y_i > 0 \\ (2^{32} - 209)/(2^{32} - 208) & \text{if } Y_i = 0 \end{cases} \end{aligned}$$

- This generator is actually pretty simple, works well, and has an amazing cycle length of about 2^{191} and excellent randomness properties!

The RNG Streams Package of L'Ecuyer et al. (2002)

This object-oriented package splits the entire cycle of the combined generator MRG32k3a in *nonoverlapping* streams, each of length 2^{127}.

- Every stream is divided into may *nonoverlapping* substreams, each of lenth 2^{51}.
- Within a stream, replication 1 uses substream 1, replication 2 uses substream 2, etc. Jumping between substreams is very fast.
- This guarantees the (approximate) independence of statistics obtained from different replications.

Outline

Random Numbers in Simio: The Mersenne Twister RNG

Simio uses an implementation of the Mersenne Twister RNG www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html with a period of $2^{19937} - 1$.

The cycle is divided into many nonoverlapping streams (subsequences). The default stream is 0.

Each stream is comprised of many substreams, each with length $2^{256} \approx 10^{77}$.

Experiments: Within a given stream, replication 1 uses substream 1, replication 2 uses substream 2, etc. The starting points for each substream are chosen at random.

Since the period of the generator is so large, there is an extremely small chance for overlapping random numbers in different replications. For example, if we randomly select 1,000,000 substreams of length 10^{77}, the probability of overlapping between the substreams will be $< 10^{-5910}$ ...

Outline

Choosing a Good Generator — Some Theory

Here's a theorem that gives a condition for multiplicative generators to attain full period.

Theorem: The multiplicative generator $X_i = aX_{i-1} \bmod m$, with prime m has full period $(m-1)$ if and only if

(a) m divides $a^{m-1} - 1$.

(b) For all integers $i < m-1$, m does not divide $a^i - 1$.

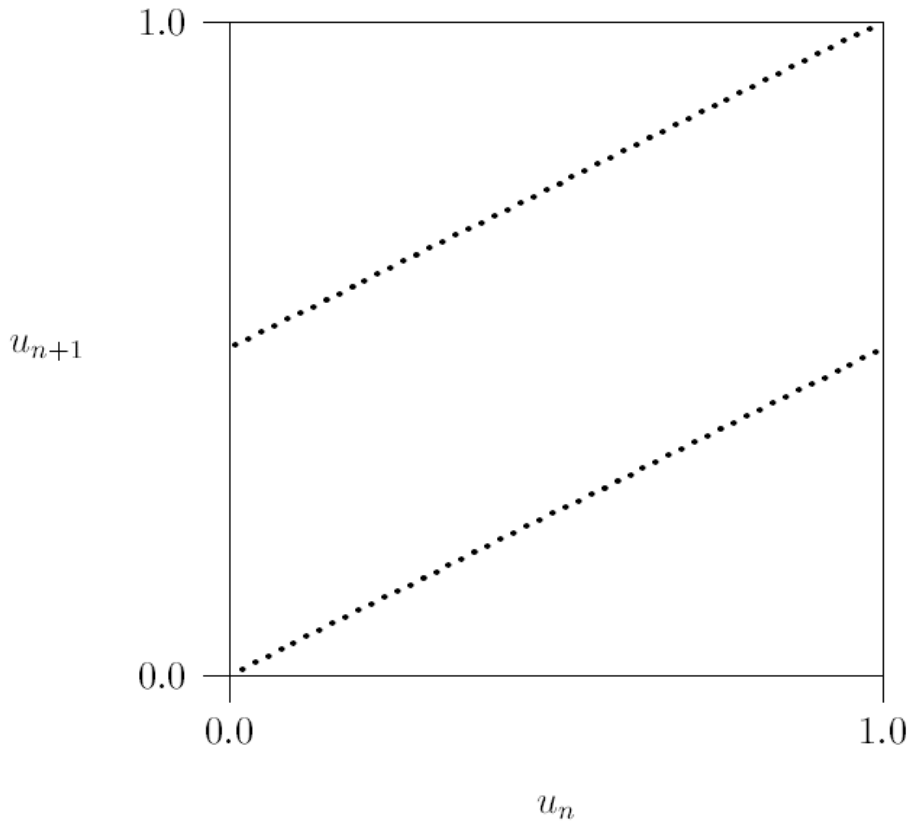
How many such multipliers exist?

For $m = 2^{31} - 1$, it can be shown that 534,600,000 multipliers yield full period.

Remark: The "best" multiplier with $m = 2^{31} - 1$ is $a =$ 950,706,376 (Fishman and Moore 1986).

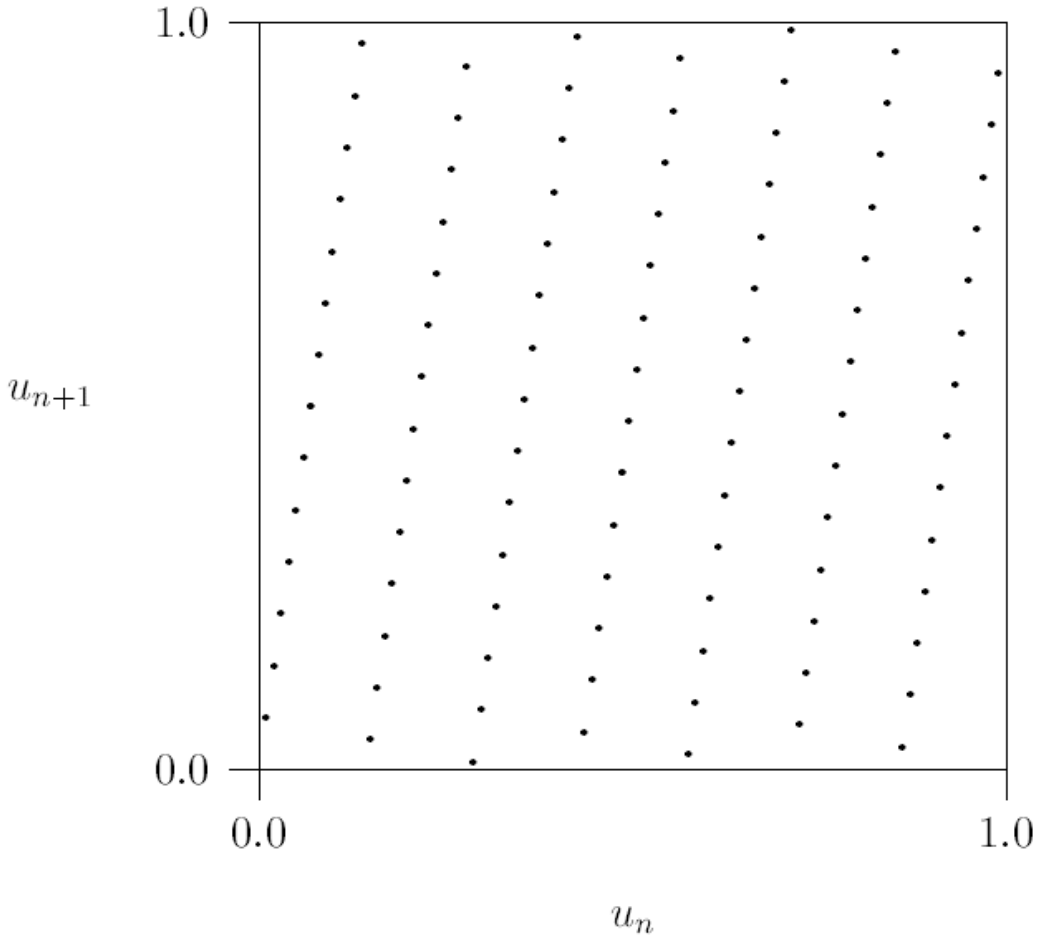
Geometric Considerations

Theorem: The k-tuples $(U_i, \ldots, U_{i+k-1})$, $i \geq 1$, from multiplicative generators lie on parallel (hyper)planes in $[0, 1]^k$.



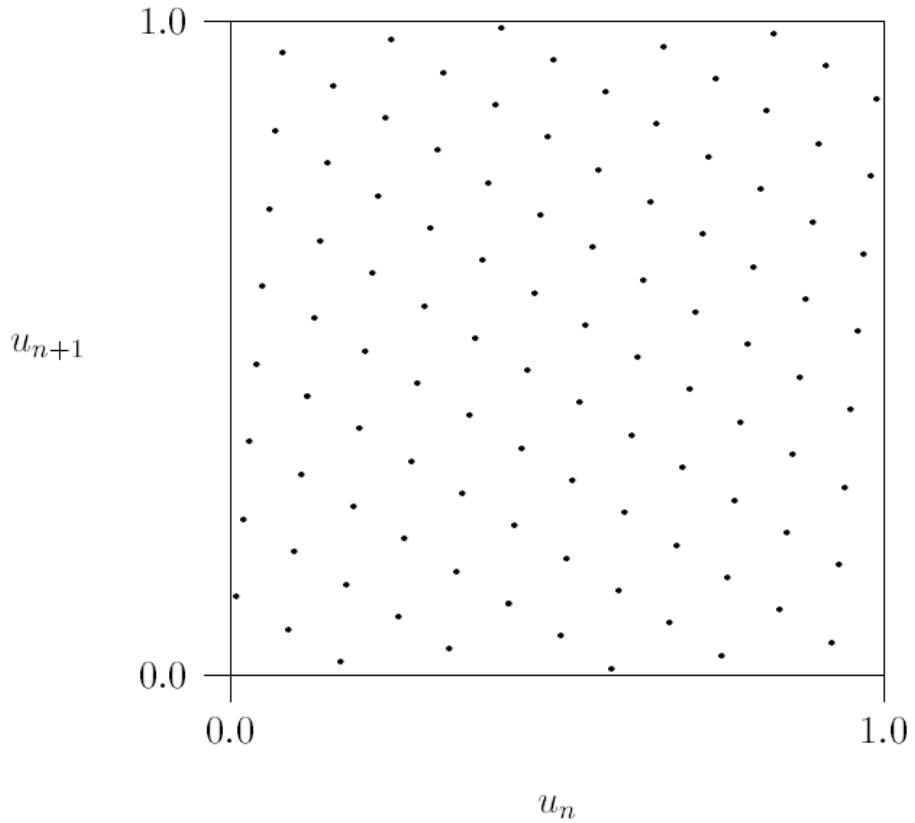
All pairs (u_n, u_{n+1}) for the LCG with $m = 101$ and $a = 51$.

Geometric Considerations (illustration)



All pairs (u_n, u_{n+1}) for the LCG with $m = 101$ and $a = 7$.

Geometric Considerations (illustration)



All pairs (u_n, u_{n+1}) for the LCG with $m = 101$ and $a = 12$.

Geometric Considerations — Theoretical Measures

The following geometric measures have been studied:

- Minimum number of hyperplanes (in all directions). Find the multiplier that maximizes this number.
- Maximum distance between parallel hyperplanes. Find the multiplier that minimizes this number.
- Minimum Euclidean distance between adjacent k-tuples. Find the multiplier that maximizes this number.

Remark: The RANDU generator is particularly bad since it lies on only 15 hyperplanes.

Outline

Choosing a Good Generator — Statistical Tests

We'll look at two classes of tests:

Goodness-of-fit tests — are the PRN's approximately Unif(0,1)?

Independence tests — are the PRN's approximately independent?

If a particular generator passes both types of tests (in addition to other tests that we won't tell you about), we'll be happy to use the PRN's it generates.

All tests are of the form H_0 (our null hypothesis) vs. H_1 (the alternative hypothesis).

We regard H_0 as the status quo, so we'll only reject H_0 if we have "ample" evidence against it.

In fact, we want to avoid incorrect rejections of the null hypothesis. Thus, when we design the test, we'll set the *level of significance*

$$\alpha \equiv P(\text{Reject } H_0 | H_0 \text{ true}) = P(\text{Type I error})$$

(typically, α = 0.05 or 0.1). We won't worry about Type II error at this point.

χ^2 Goodness-of-Fit Test

Test $H_0 : U_1, U_2, \ldots U_n \sim$ Unif(0,1).

Divide the unit interval into k cells (subintervals). If you choose equi-probable cells $[0, \frac{1}{k}), [\frac{1}{k}, \frac{2}{k}), \ldots, [\frac{k-1}{k}, 1]$, then a particular observation U_j will fall in a particular cell with prob $1/k$.

Tally how many of the n observations fall into the k cells. If $O_i \equiv$ # of R_j's in cell i, then (since the U_j's are iid), we can easily see that $O_i \sim$ Bin$(n, \frac{1}{k})$, $i = 1, 2, \ldots, k$.

Thus, the expected number of U_j's to fall in cell i will be $E_i \equiv \mathsf{E}[O_i] = n/k$, $i = 1, 2, \ldots, k$.

We'll reject the null hypothesis H_0 if the O_i's don't match well with the E_i's.

The χ^2 goodness-of-fit statistic is

$$\chi_0^2 \equiv \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}.$$

A large value of this statistic indicates a bad fit.

In fact, we *reject* the null hypothesis H_0 (that the observations are uniform) if $\chi_0^2 > \chi_{k-1,1-\alpha}^2$, where $\chi_{k-1,1-\alpha}^2$ is the appropriate $(1-\alpha)$ quantile from a χ^2 table, i.e., $P(\chi_{k-1}^2 < \chi_{k-1,1-\alpha}^2) = 1-\alpha$.

If $\chi_0^2 \leq \chi_{k-1,1-\alpha}^2$, we *fail to reject* H_0.

Usual recommendation from baby stats class: For the χ^2 GOF test to work, pick k, n such that $E_i \geq 5$ and n at least 30. But...

Unlike what you learned in baby stats class, when we test PRN generators, we usually have a *huge* number of observations n (at least millions) with a large number of cells k. When k is large, we can use the approximation

$$\chi^2_{k-1,1-\alpha} \approx (k-1)\left[1 - \frac{2}{9(k-1)} + z_{1-\alpha}\sqrt{\frac{2}{9(k-1)}}\right]^3,$$

where $z_{1-\alpha}$ is the respective standard normal quantile.

Illustrative Example: $n = 100$ observations, $k = 10$ intervals. Thus, $E_i = 10$ for $i = 1, 2, \ldots, 10$. Further, suppose that $O_1 = 13$, $O_2 = 8$, $\ldots$, $O_{10} = 11$. (In other words, 13 observations fell in the cell [0,0.1), etc.)

Turns out that

$$\chi_0^2 \equiv \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} = 3.4.$$

Let's take $\alpha = 0.05$. Then from χ^2 tables, we have

$$\chi^2_{k-1,1-\alpha} = \chi^2_{9,0.95} = 16.9.$$

Since $\chi_0^2 < \chi^2_{k-1,1-\alpha}$, we fail to reject H_0, and so we'll assume that the observations are approximately uniform.

Sources

The site below contains a plethora of reliable sources:

http://simul.iro.umontreal.ca