Randomness

1. True randomness

True intrinsic randomness is found in Quantum Mechanics.

Is it found anywhere else? Classical Mechanics? (Poincaré)
Physicists know quantum mechanics, but most statisticians do not,
so they have no reliable model for randomness.

The statistician's expression for random is i.i.d. .
(independent and identically distributed)

There is often confusion between what is unknown and what is random.
We need a good (operational) definition of randomness. [Later ...]
A good definition could help us to make a good random number generator.

It is possible to make a random number generator for computers based on physical processes that are truly random, but they are slow, inconvenient to use and sometimes inaccurate (biased).

2. Pseudorandomness.

In the 1940's, John von Neumann and friends were amazed to discover that simple arithmetic operations like integer multiplication could produce sequences of numbers that appeared to be random. They called them pseudorandom. Nobody understood why they were (almost) random.

Around 1960 the first computers became available in universities.
They came with a Fortran compiler and a small library of subroutines like sine, cosine, logarithms, and ... a Random Number Generator.
Everyone assumed that all these things made by IBM worked correctly.

But our MC calculations sometimes gave funny results: lots of regular peaks in a distribution that should be smooth.
We complained, IBM sent us a new library, and everything seemed OK.

In 1968, George Marsaglia published his famous paper explaining that the problem we saw was inherent in all multiplicative congruential generators (the kind everybody was using).

3. Pseudorandomness: "La fuite en avant": [Running away forwards]

The following years produced progress in finding defects in existing generators, and generators with longer periods, but no progress in explaining why the numbers were random.

The situation is well summarized in the book of Donald Knuth, The Art of Computer Programming, Vol. 2, 3rd edition, 1994. Over 600 pages on pseudorandom number generators, but no hint about how they make randomness. A lot about how hard it is to define.

All the experts were number theorists. Number theory is not very random.

The knowledge was that a long period was necessary, but not sufficient to guarantee randomness. One by one, all the known algorithms were found to have defects which make them unsuitable for many MC problems.

The state of the art was represented by RCARRY, a.k.a. AWC or SWB, the algorithm of Marsaglia and Zaman (1991).

4. RCARRY (Marsaglia and Zaman, 1991)

The state of the generator at a given step is given by:

$$\text{an array } X \text{ of 24 numbers with 24-bit mantissas.}$$
$$\text{and 2 indices } 1 \leq (r, s) \leq 24 \text{ , with starting values } r = 10, s = 24$$

At each step, the next random number is calculated by:

$$r = r + 1 \mod 24, \quad s = s + 1 \mod 24.$$
$$Y = |X(r) - X(s)| \text{ is the next RN, and } Y \text{ replaces } X(s).$$

[There are some other details, not of interest here.]
Apart from index manipulation, there is only one arithmetic operation.
But the number of possible states of the array A is $(2^{24})^{24}$.
In fact, the period is 1/48 of this number, approximately 5×10^{171}.

Another popular RNG is the Mersenne Twister, based on an array in memory of length 624 words. It is even faster than RCARRY and has a much longer period.

5. Two physicists from Armenia

I received one day a visit from two Armenian physicists (Akopov and Savvidy) who said the right way to generate independent sequences was to make use of Kolmogorov mixing (not related to Kolmogorov complexity).

Representing the state of the generator at the i^{th} step by the contents of an array $\mathbf{X}_i$, their generator proceeds to the next step by performing the linear transformation

$$\mathbf{X}_{i+1} = \mathcal{A}\,\mathbf{X}_i$$

and they said that to obtain Kolmogorov mixing, the matrix $\mathcal{A}$ had to have

- Determinant $= 1$
- All eigenvalues are complex and distinct
- No eigenvalue has modulus 1

I didn't understand any of this, so I thought I would first try to find out if the generator seemed to work. Unfortunately, it didn't.

6. The Crisis in Ising Model Simulations

Meanwhile a crisis was developing in Condensed Matter Physics where the "best" generators were giving wrong results. The failures were being revealed by a new algorithm invented by Ulli Wolff to overcome the critical slowing down that had made it nearly impossible to simulate across phase transitions.
[Ferrenberg, Landau and Wong, MC Simulations: Hidden errors from "good" random number generators, PRL 69, 3382-3384 (1992)]

Ulli Wolff told me his algorithm worked by flipping many spins at once, instead of one at a time, so he thought it would be more sensitive to long-term correlations in the random number generator than other algorithms.
He also said there was a theoretician at DESY, Martin Lüscher, who had found a way to make a better generator, but it was coded only for the special-purpose computer APE designed for Lattice Gauge calculations.

7. Lüscher's algorithm

To my great surprise, Martin Lüscher had decided to base his new generator on the RCARRY algorithm. This was surprising because RCARRY was already known to be defective, but Martin knew exactly what he was doing.

If you consider the RCARRY algorithm as a sequence of points in 24-dimensional space, then it is of the form

$$\mathbf{X}_{i+1} = \mathcal{A}\, \mathbf{X}_i$$

where $\mathbf{X}_i$ is the array of 24 24-bit numbers that determines (along with the carry bit) the state of the generator.

The algorithm is then defined by the matrix $\mathcal{A}$, which you can readily see is a matrix with mostly zeroes, with 24 values "$+1$" along the diagonal and 24 "-1" at the positions corresponding to the values of r and s. The determinant is 1, all of its eigenvalues are different and none of them have modulus 1.

8. RCARRY has Kolmogorov Mixing

He had seen that the RCARRY algorithm was in fact the discrete analog of a continuous classical dynamical system with Kolmogorov mixing. And he knew what that meant:

1. Zero-mixing is the same as ergodic motion. It means that the system will asymptotically sweep out all the available states.
2. One-mixing means that the coverage will be uniform, in the sense that asymptotically, the probability of finding the system in any given volume of state space is proportional to the volume.
3. Two-mixing means that the probability of the system being in one volume of state space at one time, and another volume at another time, is proportional to the product of the two volumes.
4. N-mixing means that the probability of finding the system in N different volumes at N different times is proportional to the product of the N volumes.
5. k-mixing is N-mixing for arbitrarily large N.

9. Lüscher's algorithm: The Lyapunov Exponent

Now the big question: Why does RCARRY fail the tests of randomness?

Answer: Because Kolmogorov mixing is only an asymptotic property of classical dynamical systems.

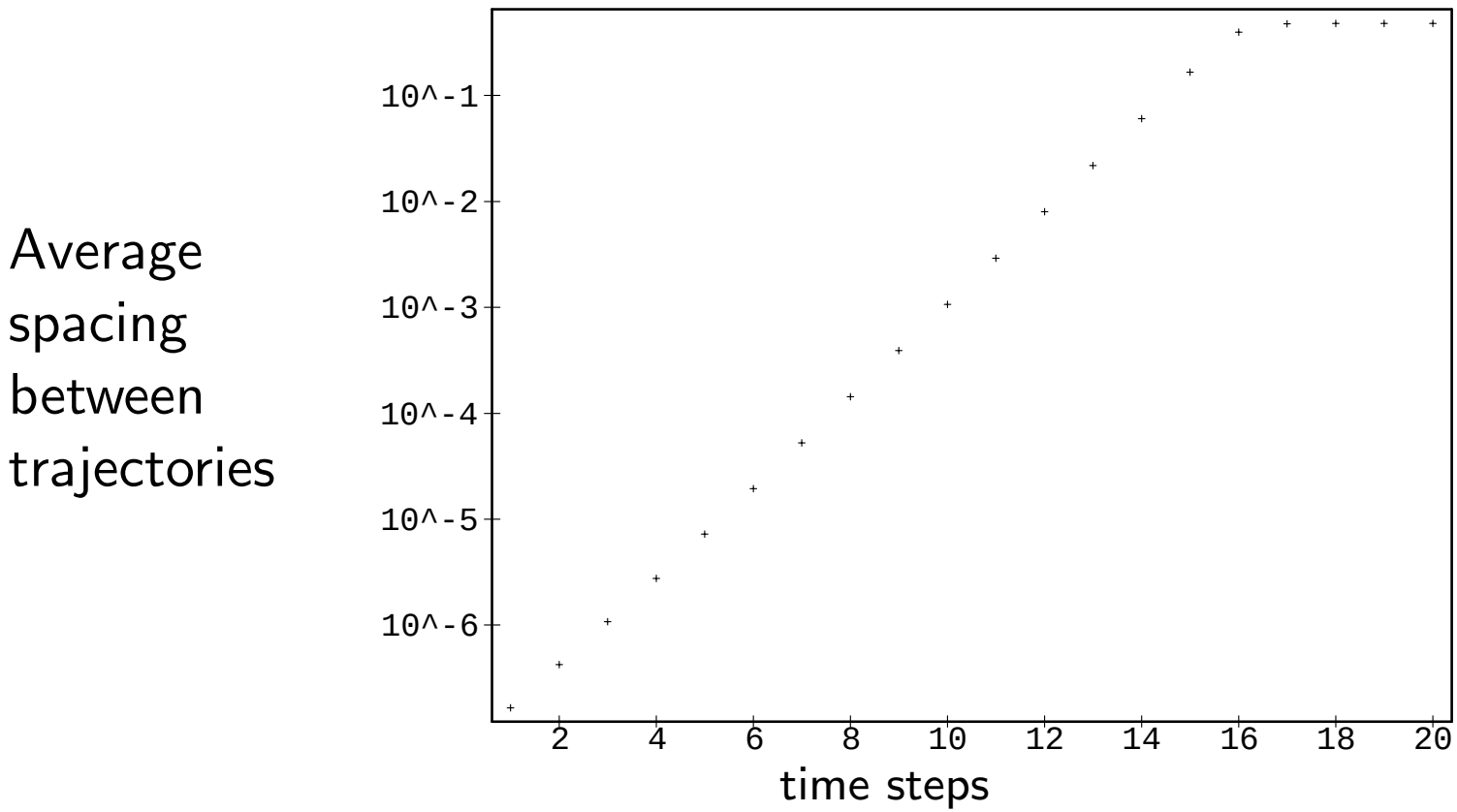
K-systems diverge exponentially, in the following sense:
If a k-system is initiated from two different but arbitrarily close initial states, the two trajectories will diverge exponentially. The exponent describing the speed of divergence is called the Lyapunov exponent.

For the matrix $\mathcal{A}$, there are 4 eigenvalues with maximum absolute value $|\lambda|_{max} = 1.04299..$, and the Lyapunov exponent is $\ln |\lambda|_{max} = 1.01027..$

For this system, that means that full mixing (where the deviation between the trajectories exceeds $1/2$) is attained only after 16 applications of $\mathcal{A}$.

10. Lüscher's algorithm: Decimation

This figure shows the "experimental" divergence of nearby trajectories after the number of time steps shown on the x-axis.



After generating 24 numbers, you must reject 15×24 numbers before using the next 24, in order to attain complete mixing, or independence.

M. Lüscher, Comp. Phys. Comm. 79 (1994) 100

11. RANLUX: the Luxury Level

When I coded Lüscher's algorithm in Fortran, I called it RANLUX because it offered several different luxury levels:

1. Level Zero with no decimation,corresponds to RCARRY: fast, and good enough for many applications, but known to have easily detectable defects.
2. Level One with half the numbers rejected, is almost as fast as RCARRY, but with considerably better randomness.
3. Level Two with acceptance 1/4 is about half the speed of RCARRY, and much more random, but still has detectable defects.
4. Level Three with acceptance 1/8 has theoretically detectable defects, but none has yet been detected.
5. The Highest Level, corresponding to rejecting 15 steps, should have no defects detectable by any possible tests. Unfortunately it is about 8 times slower than RCARRY.

The idea was that if you can afford the luxury of a provably good PRNG, you can use RANLUX at the highest luxury level. Otherwise you should use the highest luxury level you can afford.

12. Continuous and discrete systems

The major theoretical problem with Lüscher's algorithm is that the Kolmogorov k-systems are continuous dynamical systems, but we are applying the theory to discrete systems.

An important difference between discrete and continuous systems is that a continuous system can follow an infinitely long trajectory in state space without ever encountering a previously occupied state.

A discrete system, on the other hand, has only a finite number of possible states, so it eats up its state space as it moves through it, in the sense that all those states it has already visited are removed from the space of states it has available for future visits. Otherwise it gets into a loop.

However it is easily seen that, since the period of RANLUX is $\approx 10^{171}$, even if it was used to generate all the random numbers that have ever been generated plus all those that will ever be generated on the earth, it would not eat up more than 10^{-130} of its total state space.

13. The speed of RANLUX

Many people consider RANLUX to be too slow. These people don't seem to be worried about getting the right answer to their calculation. Marsaglia himself once stated (in a published paper):

> *Random numbers are like sex: Even if they are not very good, they are still good.*

I don't understand this attitude, so I published my viewpoint on the matter:

> *Random numbers are more like wine: It is better to serve one that is too good, than one that is not good enough.*

Even if some consider RANLUX too slow, most physicists are happy that they finally have access to a provably good PRNG.

Now that the Mersenne Twister is known to fail some tests,
all the most popular PRNG are known to have defects
except Ranlux at the highest luxury level.

14. Progress with MIXMAX

After my first conversation with Akopov and Savvidy, they continued working on MIXMAX, reducing the computation time from $O(N^2)$ to $O(N \ln N)$ and finding matrices with optimal mixing properties and generators with long periods.

More recently, George Savvidy's son Konstantin Savvidy has continued this work and has reduced the computation time to $O(N)$, which finally makes it comparable in speed with other generators.

Just two weeks ago, Konstantin has submitted a new version of MIXMAX to HEPFORGE, so it is now an available working generator.

The default matrix size is now 32×32, and the generator produces extended precision reals or 60-bit integers. A lot of work has gone into assuring long periods and good mixing (eigenvalues as far as possible from the unit circle to obtain a big Lyapunov exponent and avoid decimation).

15. Conclusion: How to decide which RNG to use?

Apart from some MC applications which are known to be very sensitive to the quality of the random numbers, most MC users have no way to judge how good their random numbers have to be.

It is now known that all the fast, popular RNG's have serious defects.
In view of the history of RN generation, it is very likely that these RNG's also have other defects not yet known.
This is not surprising, since there is no known reason why any of them should produce random numbers at all.

The problem facing the MC user is whether these defects will cause him to get incorrect results from his calculation/simulation. MC results have the unfortunate property that it is often very hard to know if they are correct. If they are wrong because of a defective RNG, it is even harder to detect.

16. Solution: Use an RNG that produces provably random numbers.

The only theory of randomness that has been applied to RNG is that of Kolmogorov, Anosov, Sinai, Arnold, Rohlin, and others developed mostly in the Soviet Union in the 1950's and 60's to describe the behaviour of classical mechanical systems which are not solvable analytically.
We associate randomness with the idea of Kolmogorov Mixing, such that the most random systems are the Kolmogorov k-systems.

RANLUX

Martin Luscher's RANLUX, at the highest luxury level, has never failed any tests of randomness, but it is about ten times slower than the Mersenne Twister. It has been used successfully for many years by many groups.

MIXMAX

The Savvidy family's MIXMAX makes optimal use of the Kolmogorov theory to produce a RNG which should mix faster than RANLUX, but the implementation is new and we have no experience with it.