

RANDOM NUMBER GENERATORS ARE CHAOTIC

Charles Herring

U.S. Army Construction
Engineering Research Laboratory
P.O. Box 4005
Champaign, IL 61824

c-herring@cecer.army.mil

Julian I. Palmore

University of Illinois
Department of Mathematics
1409 W. Green Street
Urbana, IL 61801

j-palmore@cecer.army.mil

ABSTRACT

We observe that pseudo-random number generators, familiar to all programmers, are derived from deterministic chaotic dynamical systems. We discuss the implications of this finding and compare computer generation of pseudo-random numbers to the theoretical ideal of a (noncomputable) random sequence.

1. INTRODUCTION

The study of highly unstable nonlinear dynamical systems--chaotic systems--has emerged recently as an area of major interest and applicability across the mathematical, physical and social sciences. This interest has been triggered by advances in the past decade, particularly in the mathematical understanding of complex systems. An important insight that has become widely recognized in recent years is that deterministic systems can give rise to chaotic behavior. Surprisingly, many of these systems are extremely simple, yet they exhibit complex chaotic behavior.

Of major interest to the programming community is the design and maintenance of standard pseudo-random number generators that are portable and reusable. The design of pseudo-random number generators and their computer implementation has a long history. The modern aspects of this problem began with the theory of computability (Church and Turing in the late 1930s) and with the design and construction of electronic computing machines (von Neumann). More recent work (Chaitin, Kolmogorov and Martin-Lof) has provided a firm theoretical basis for understanding randomness.

In this article we observe that pseudo-random number generators, familiar to all programmers, are derived from deterministic chaotic dynamical systems. As we shall see, when pseudo-random number generators are designed properly, the sequences produced are completely predictable.

2. RANDOM SEQUENCES

Random number generation has been of great interest from the beginnings of computing. Philosophically and mathematically, the concept of randomness poses many problems. An interesting historical account of the definitions proposed for random sequences is found in Knuth [3]. Intuitively, we equate randomness with unpredictability.

In the past 25 years, largely due to the separate efforts of Chaitin [1] and Kolmogorov the concept of randomness has been made definitive. They accomplished this by developing the concept of algorithmic complexity. The complexity of an n-digit sequence is the length in bits of the shortest computer program that can produce the sequence. For a very regular sequence, 1111111111, for example, a very small program is needed. As the sequence becomes highly irregular and as the length of the sequence grows beyond bounds, it can be shown that the shortest program needed to produce the sequence is slightly larger than the sequence itself.

This observation leads to the definition of randomness in terms of algorithmic complexity: a sequence is random if the smallest program needed to produce it is approximately equal to the size of the sequence itself. Such a sequence is called "maximally complex". Thus, no finite algorithm can produce a maximally complex sequence of infinite length. A consequence is that truly random sequences cannot be computed. Additionally, Martin-Lof [5], through a statistical theory approach, has proven that a random sequence must pass all of a countable infinity of statistical tests.

Clearly, any algorithmic implementation of the theoretical ideal of randomness on a computer will be imperfect. From all view points above, a random sequence is a noncomputable (and therefore nonperiodic) infinite sequence. A measure of the "goodness" of a pseudo-random number generator is aperiodicity. However, any finite computer algorithm implementation yields only periodic sequences--although of very long period. Thus, random number generators found in computer languages are referred to as "pseudo-random" number generators.

In 1951, Lehmer [4] proposed an algorithm that has become the de facto standard pseudo-random number generator. As it is usually implemented, the algorithm is known as a Prime Modulus Multiplicative Linear Congruential Generator. It is better known as a Lehmer generator. The form of the Lehmer generator is:

$$f(x_n) = x_{n+1} = a x_n \bmod m.$$

Park and Miller [7] propose a portable minimal standard Lehmer generator. Their choice of m , the prime modulus, is $2^{31} - 1$, and their choice of the multiplier is $a = 7^5$. Their minimal standard Lehmer generator is given by:

$$f(x) = 16807 x \bmod 2147483647.$$

Park and Miller give the Pascal implementation for four versions of this generator and

discuss the precision requirements. For example, their real number version requires that reals have at least a 46-bit mantissa. This specification allows the minimal standard Lehmer generator to be ported to any machine that can meet this requirement. There is much programming lore regarding the choice of the initial seed, \mathbf{x}_1 , and for some generators this is a concern. Park and Miller point out that their generator allows any choice of seed between $\mathbf{1}$ and $\mathbf{m} - \mathbf{1}$ because the generator is full-period.

3. CHAOTIC DYNAMICAL SYSTEMS

In everyday usage, chaos is synonymous with great disorder. Gleick [2], in his popular book on chaos, gives the views of several researchers:

- (i) the complicated, aperiodic attracting orbits of certain dynamical systems,
- (ii) apparently random recurrent behavior in a simple deterministic system,
- (iii) the irregular, unpredictable behavior of deterministic nonlinear dynamical systems.

It follows that qualitatively that the behavior of chaotic systems is nonperiodic, greatly disordered, deterministic--yet apparently unpredictable and random. In current usage, a system is chaotic if it has sensitive dependence on initial conditions. Associated with this sensitive dependence is a geometric growth in small differences with time.

The simplest chaotic dynamical system is the Bernoulli shift described by Palmore [6]:

$$\mathbf{x}_{n+1} = \mathbf{D} \mathbf{x}_n \bmod \mathbf{1},$$

where \mathbf{D} is an integer larger than $\mathbf{1}$. In base \mathbf{D} arithmetic, \mathbf{x} is written as:

$$\mathbf{x} = .\mathbf{a}_1 \mathbf{a}_2 \dots ,$$

where \mathbf{a}_i takes values $[\mathbf{0}, \mathbf{1}, \dots, \mathbf{D}-\mathbf{1}]$.

Thus, the mapping in base \mathbf{D} arithmetic is a shift of the base point to the right by one digit with each application and $\bmod \mathbf{1}$ recovers the fractional part. If we let $\mathbf{D} = \mathbf{2}$ (base 2), the initial condition \mathbf{x}_0 is a binary number. Each iteration of the Bernoulli shift returns a binary number where each bit \mathbf{a}_i can be thought of as being the "heads" or "tails" of a coin toss experiment. This dynamical system on the interval exhibits sensitive dependence on initial conditions (\mathbf{x}_0) and is characterized by a geometric growth of \mathbf{D} with each iteration.

The Bernoulli shift is a simple algorithm that exhibits complex chaotic behavior. It is deterministic in that it is a specific algorithm implemented by a definite set of instructions in a computer language on a computer with finite precision. It is characterized by exponential growth and disorder. It has sensitive dependence on initial conditions.

4. RANDOM NUMBER GENERATORS AS CHAOTIC DYNAMICAL SYSTEMS

Now we are ready to make the connection between chaos and pseudo-random number generators. We have seen that the Bernoulli shift is a deterministic chaotic process. By inspection, it is similar in form to the Lehmer generator. With the example of the Bernoulli shift above, we observe that prime modulus multiplicative linear congruential generators are implementations of deterministic chaotic processes. They are Bernoulli shifts on integers. The multiplier a and the modulus m are chosen with great care to ensure that a full period is produced. A specific implementation of a full period generator is a single cycle system. Seeding starts the generator at a given point on the cycle--the entire sequence will be produced if the generator is called m times. This algorithm depends sensitively on the choice of multiplier and modulus.

In fact, all pseudo-random number generators, which are shifts on finite precision integers, have a definite but very long period. Thus, a pseudo-random number sequence is a reordering of the integers from 1 to $m-1$. The success of such a generator is in achieving maximum disorder as measured by the complexity of the reordered integers.

5. REFERENCES

1. Chaitin, G.J., "Randomness and mathematical proof," *Sci. Amer.* 232 (1975), 47.
2. Gleick, J., *Chaos: making a new science* (Viking, New York, 1987), 306.
3. Knuth, D. E., *The Art of Computer Programming, Vol 2, Seminumerical Algorithms*, 2nd Ed. (1981).
4. Lehmer, D.H., "Mathematical methods in large-scale computing units," *Ann. Computing Lab. Harvard Univ.* 26 (1951), 141-146.
5. Martin-Lof, P., "The Definition of Random Sequences," *J. Inform Control*, 9 (1966), 602.
6. Palmore, J.I., and McCauley, J.L., "Shadowing by computable chaotic orbits," *Phys. Lett. A* 121, (June 1987), 399.
7. Park, S.K., and Miller, K.W., "Random number generators: good ones are hard to find," *Comm. ACM* 31, 10 (October 1988), 1192-1201.