Chapter 7. Random Numbers

7.0 Introduction

It may seem perverse to use a computer, that most precise and deterministic of all machines conceived by the human mind, to produce “random” numbers. More than perverse, it may seem to be a conceptual impossibility. Any program, after all, will produce output that is entirely predictable, hence not truly “random.”

Nevertheless, practical computer “random number generators” are in common use. We will leave it to philosophers of the computer age to resolve the paradox in a deep way (see, e.g., Knuth [1] §3.5 for discussion and references). One sometimes hears computer-generated sequences termed *pseudo-random*, while the word *random* is reserved for the output of an intrinsically random physical process, like the elapsed time between clicks of a Geiger counter placed next to a sample of some radioactive element. We will not try to make such fine distinctions.

A working, though imprecise, definition of randomness in the context of computer-generated sequences, is to say that the deterministic program that produces a random sequence should be different from, and — in all measurable respects — statistically uncorrelated with, the computer program that *uses* its output. In other words, any two different random number generators ought to produce statistically the same results when coupled to your particular applications program. If they don’t, then at least one of them is not (from your point of view) a good generator.

The above definition may seem circular, comparing, as it does, one generator to another. However, there exists a body of random number generators which mutually do satisfy the definition over a very, very broad class of applications programs. And it is also found empirically that statistically identical results are obtained from random numbers produced by physical processes. So, because such generators are known to exist, we can leave to the philosophers the problem of defining them.

A pragmatic point of view, then, is that randomness is in the eye of the beholder (or programmer). What is random enough for one application may not be random enough for another. Still, one is not entirely adrift in a sea of incommensurable applications programs: There is a certain list of statistical tests, some sensible and some merely enshrined by history, which on the whole will do a very good job of ferreting out any correlations that are likely to be detected by an applications program (in this case, yours). Good random number generators ought to pass all of these tests; or at least the user had better be aware of any that they fail, so that he or she will be able to judge whether they are relevant to the case at hand.

As for references on this subject, the one to turn to first is Knuth [1]. Then try [2]. Only a few of the standard books on numerical methods [3-4] treat topics relating to random numbers.

CITED REFERENCES AND FURTHER READING:

Knuth, D.E. 1981, *Seminumerical Algorithms*, 2nd ed., vol. 2 of *The Art of Computer Programming* (Reading, MA: Addison-Wesley), Chapter 3, especially §3.5. [1]

Bratley, P., Fox, B.L., and Schrage, E.L. 1983, *A Guide to Simulation* (New York: Springer-Verlag). [2]

Dahlquist, G., and Bjorck, A. 1974, *Numerical Methods* (Englewood Cliffs, NJ: Prentice-Hall), Chapter 11. [3]

Forsythe, G.E., Malcolm, M.A., and Moler, C.B. 1977, *Computer Methods for Mathematical Computations* (Englewood Cliffs, NJ: Prentice-Hall), Chapter 10. [4]

7.1 Uniform Deviates

Uniform deviates are just random numbers that lie within a specified range (typically 0 to 1), with any one number in the range just as likely as any other. They are, in other words, what you probably think "random numbers" are. However, we want to distinguish uniform deviates from other sorts of random numbers, for example numbers drawn from a normal (Gaussian) distribution of specified mean and standard deviation. These other sorts of deviates are almost always generated by performing appropriate operations on one or more uniform deviates, as we will see in subsequent sections. So, a reliable source of random uniform deviates, the subject of this section, is an essential building block for any sort of stochastic modeling or Monte Carlo computer work.

System-Supplied Random Number Generators

Most C implementations have, lurking within, a pair of library routines for initializing, and then generating, "random numbers." In ANSI C, the synopsis is:

```
#include <stdlib.h>
#define RAND_MAX ...

void srand(unsigned seed);
int rand(void);
```

You initialize the random number generator by invoking `srand(seed)` with some arbitrary `seed`. Each initializing value will typically result in a different random sequence, or a least a different starting point in some one enormously long sequence. The *same* initializing value of `seed` will always return the *same* random sequence, however.

You obtain successive random numbers in the sequence by successive calls to `rand()`. That function returns an integer that is typically in the range 0 to the largest representable positive value of type `int` (inclusive). Usually, as in ANSI C, this largest value is available as `RAND_MAX`, but sometimes you have to figure it out for yourself. If you want a random `float` value between 0.0 (inclusive) and 1.0 (exclusive), you get it by an expression like