CHAPTER 12

Large Random Numbers

> Mathematics is full of pseudorandomness—plenty enough to supply all would-be creators for all time.
>
> —D. R. Hofstadter, *Gödel, Escher, Bach*

> Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.
>
> —John von Neumann,

SEQUENCES OF "RANDOM" NUMERICAL VALUES are used in many statistical procedures, in numerical mathematics, in physics, and also in number-theoretic applications to replace statistical observations or to automate the input of variable quantities. Random numbers are used:

- to select random samples from a larger set,
- in cryptography to generate keys and in running security protocols,
- as initial values in procedures to generate prime numbers,
- to test computer programs (a topic to which we shall return),
- for fun,

as well as in many additional applications. In computer simulations of natural phenomena random numbers can be used to represent measured values, thereby representing a natural process (*Monte Carlo methods*). Random numbers are useful even when numbers are required that can be selected arbitrarily. Before we set out in this chapter to produce some functions for the generation of *large* random numbers, which will be required, in particular, for cryptographic applications, we should take care of some methodological preparations.

There are many sources of random numbers, but we should be sure to differentiate between *genuine random numbers*, which arise as the result of random experiments, and *pseudorandom numbers*, which are generated algorithmically. Genuine random numbers arise from such processes as the tossing of coins or dice, spinning a (fair) roulette wheel, observing processes of radioactive decay with the aid of suitable measuring equipment, and evaluating the output of electronic components. In contrast to these, pseudorandom

numbers are computed by algorithms, generated with the aid of *pseudorandom number generators*, which are deterministic, in that they depend only on an initial state and initial value (seed), and therefore are both predictable and reproducible. Pseudorandom numbers thus do not arise *randomly* in the strict sense of the word. The reason that this situation can frequently be ignored is that we are in possession of algorithms that are able to produce pseudorandom numbers of "high quality," where we shall have to explain what we mean by this term.

The first thing that we establish is that in fact, it makes no sense to talk about a single number being "random," but that mathematical requirements for randomness are always satisfied by *sequences* of numbers. Knuth speaks of a *sequence of independent random numbers with a particular distribution*, in which every number is produced *randomly and independently of all other numbers of the sequence*, and every number assumes a value within a certain range of values with a certain probability (see [Knut], Section 3.1). We use the terms "random" and "independent" here to mean that the events leading to the selection of concrete numbers are too complex in their formation and interaction to be detected by statistical or other tests.

This ideal is theoretically unachievable by generating numbers using deterministic procedures. Yet the goal of many different algorithmic techniques is to approach this ideal as closely as possible. The logical structure of deterministic random number generators can be described by a quintuple $(S, R, \phi, \psi, P_{\text{start}})$, where S denotes the finite set of internal states of the generator, R is the set of possible output values, $\phi : S \to S$ is the state function, $\psi : S \to R$ is the output function, and P_{start} is a probability measure for the distribution of the initial state s_0. After initialization, at each step $n \geq 1$, first the new state $s_n := \phi(s_{n-1})$ is computed, and from this state, the output value $r_n := \psi(s_n)$ (see [BSI2]). For the evaluation of the random number generators that we are considering here, we begin with the assumption that the start state is uniformly distributed in the set S, and this is indicated with the notation μ_S (for P_{start}). In the next chapter we will be concerned with testing the FLINT/C functions. For this, we will be using large random numbers that do not yet need to satisfy any of the demands of cryptographic security.

Therefore, we first select from the many possibilities at hand a proven and frequently used procedure for generating pseudorandom numbers (for the sake of brevity we shall frequently drop the "pseudo" and speak simply of random numbers, random sequences, and random number generators) and spend some time with the method of *linear congruences*. Beginning with an initial value X_0 the elements of a sequence are generated by the linear recursion

$$X_{i+1} = (X_i a + b) \bmod m. \tag{12.1}$$

This procedure was developed in 1951 by D. Lehmer, and it has enjoyed considerable popularity since that time, since despite their simplicity, linear congruences can produce random sequences with excellent statistical properties,