Chapter 14. More Fortran Elements: Random Number Generators

14.1 Overview

In Chapter 6, the difference between chaotic and random sequences was discussed. A chaotic sequence is produced from deterministic equations. There is structure in a chaotic sequence as observed from a first return map, though no periodicity. A random sequence is not produced from deterministic equations but rather from random uncorrelated events such as coin flips. In this chapter, properties of random number sequences and methods for the production of random number sequences will be discussed.

14.2 Properties of Random Number Sequences

Figure 14.1 shows a random sequence of 200 numbers with values between 0 and 1. One of the properties of a random sequence is that all the numbers in the range have equal probability. In this case, the probability of a number in the sequence being in the range [0, 0.5] is 50 percent and the probability of a number in the sequence being in the range [0.5, 1] also is 50 percent.

![Figure 14.1. A sequence of 200 random numbers between 0 and 1.](image)

The expected probabilities for random sequences are valid in the limit that the number of trials is large; that is, they are statistical. For example, if a coin is flipped only 10 times, there is no guarantee that exactly five heads and five tails will be obtained. On the other hand, if a coin is flipped 1000 times, then the probability of obtaining heads and tails approaches 50 percent each. This is illustrated in Figure 14.2 for a sequence of 1000 numbers between 0 and 1. As the number of points increases, the average value should approach 0.5. If only the first 20 points are included in the sequence, the average value is 0.560. For the first 250 points, the average is 0.487, and for 1000 points, the average is 0.495. Clearly, the result is statistically converging to the expected result of 0.5. Of course, since the numbers in such a sequence are random, another set of numbers selected randomly will not yield the same average values; however, another sequence also will converge toward the expected average of 0.5.
Another property of random sequences is that numbers in the sequence are uncorrelated. As was seen in Chapter 6, one method that is used to check that the sequence is uncorrelated is to produce a first return map. A first return map is produced by plotting pairs of adjacent points in the sequence, $x_{n+1}$ as a function of $x_n$. For a periodic or chaotic sequence, the first return map shows a correlation between the points. However, for a random sequence, the first return map shows no correlation at all. Figure 14.3 gives an example of a first return map for the random sequence shown in Figure 14.1.
14.3 Generating Random Numbers

Most algorithms to create sequences of random numbers in reality create pseudo-random number
sequences. That is, if the random number generator is given the same starting value (called the
seed), then it will produce the same sequence every time. This behavior is known as a pseudo-
random sequence, and it is useful for testing the Fortran code. The properties of a pseudo-random
sequence are the same in terms of average and first return map as for a random sequence. However,
a pseudo-random sequence has a finite length; that is, after a certain number of points the sequence
repeats. For a good pseudo-random number generator, the repeating period is very large, so
perhaps $10^5$ or $10^6$ random numbers at a minimum may be generated before the sequence repeats.

The most common method for generating pseudo-random number sequences is known as
the linear congruence method, which will be discussed below. Other random number generating
algorithms exist. In fact, this area is a subject of a great deal of current research related to
encryption and code-breaking; however, more complex algorithms will not be discussed here.

14.3.1 The Modulus Function

One of the most common pseudo-random number generators, the linear congruence
method, is based on the Fortran modulus function. The modulus function in Fortran is denoted
$\text{MOD}(n, m)$, where $n$ and $m$ are integers. The modulus function returns the remainder when the
integer $n$ is divided by the integer $m$. The modulus function can be defined

$$\text{MOD}(n,m) = n - m \cdot \text{INT} \left( \frac{n}{m} \right),$$

(14.1)

where the Fortran function $\text{INT}$ is integer truncation. Some examples of the modulus function are

$$\text{MOD}(44,7) = 2 \quad \text{MOD}(3,11) = 3 \quad \text{MOD}(10,5) = 0$$

(14.2)

14.3.2 The Linear Congruence Method

The linear congruence method uses the modulus function and maps one integer, $X_n$, into
another one, $X_{n+1}$, according to the formula

$$X_{n+1} = \text{MOD}(aX_n + c, m).$$

(14.3)

In Equation (14.3), $a$, $c$, and $m$ are integer parameters that are chosen by the user. Because there are
only a finite number of integers a computer can handle, this sequence will arrive sooner or later at
the same integer and therefore the sequence must be periodic. This periodicity means that once an
integer occurs again, the whole sequence starts over again. This may not seem advantageous, but
the period can be made reasonably large for good choices of the parameters $a$, $c$, and $m$.

14.4 Built-In Random Number Generators

Most Fortran compilers, g77 included, have built-in (pseudo)random number generators. To use the built-in random number generator in g77, it first must be initialized with a call to a
subroutine SRAND,

CALL SRAND(SEED)
Here, SEED is an integer value that serves as the seed to the random number generator. Then, a random number between 0 and 1 may be selected by using the REAL function RAND; for example, to assign a random number between 0 and 1 to the variable X, the Fortran statement is

\[ X = \text{RAND}(0) \]

Here, the argument 0 means that the next random number in the sequence is called. If the argument equals 1, then the function is reinitialized using a new seed.

### 14.5 User-Supplied Random Number Generators

A simple Fortran function may be written to make use of the linear congruence method. A particularly useful choice of parameters \(a\), \(c\), and \(m\) in Equation (14.3) is

\[
a = 1366 \\ c = 150889 \\ m = 714025
\]  

(14.4)

With this choice, the pseudo-random sequence has a period that is 714025 numbers long. A Fortran function that implements this choice is

```fortran
REAL FUNCTION RANDOM(L)
IMPLICIT NONE
INTEGER L

L = MOD(1366*L+150889, 714025)
RANDOM = REAL(L)/REAL(714024)
RETURN
END
```

Please note that the last number in lines 4 and 5 differ on purpose by 1. Line 5 guarantees that the output random number is inside the interval \([0, 1]\). To generate a sequence of random numbers between 0 and 1 a starting integer \(L\) (the seed) has to be provided, which then is consecutively updated by the routine. If we would like to change the range of the uniform random numbers to \([a, b]\), we would have to compute the numbers as \((b-a)\times\text{RANDOM}(L)+a\).

### 14.6 Testing Random Number Generators

It is useful to test any random number generator used to make sure that it properly producing a random distribution. This is often done using the first return map discussed in Section 14.2. For example, consider two random number generators constructed with the linear congruence method and the following parameters:

Random Number Generator #1: \(a=1366\) \(c=150889\) \(m=714025\)

Random Number Generator #2: \(a=137\) \(c=187\) \(m=256\)
The first return maps for these two random number generators are shown in Figure 14.4. Note that the first random number generator appears to have no correlations; however, the second random number generator is correlated and shows periodic behavior. Thus, the second random number generator would not be a good choice for actual use.

![Figure 14.4](image)

**Figure 14.4.** First return maps for (a) Random Number Generator #1; (b) Random Number Generator #2.
Chapter 14 Review Problems

1. Evaluate the following expressions.
   
   (a) $\text{MOD}(23, 2)$
   
   (b) $\text{MOD}(42, 2)$
   
   (c) $\text{MOD}(2, 11)$

2. Use the MOD function and construct a segment of Fortran code to test an integer $M$ to determine whether the integer is odd or even.

3. The linear congruence method uses the expression $L = \text{MOD}(A \times L + C, M)$. Evaluate the expression for the following parameters.
   
   (a) $A=3$, $C=9$, $M=7$ with an initial $L=2$
   
   (b) $A=3$, $C=9$, $M=7$ with an initial $L=1$.
   
   How many different values of $L$ are possible as a result?

4. A potential random number generator function is shown below.

   ```fortran
   REAL FUNCTION RANDOM(L)
   IMPLICIT NONE
   INTEGER L

   L = MOD(2*L+5, 3)
   RANDOM = REAL(L)/REAL(2)

   RETURN
   END
   ```

   (a) Starting with a seed of $L=2$, what is the sequence of the first 6 numbers produced?

   (b) Starting with a seed of $L=1$, what is the sequence of the first 6 numbers produced?

   Is this a good random number generator?

5. Construct the first return maps for the two sequences of problem 4.