4.1.3 Additive Congruential Method

The additive congruential method produces a sequence of integers between 0 and $m-1$ according to the recursion

$$Z_i = (Z_{i-1} + Z_{i-k}) \bmod m, \qquad , i = 1, 2, 3, \ldots,$$

where $k \geq 2$. To obtain the desired uniformly (0,1) distributed random numbers we again should choose $U_i = Z_i/m$. The method can have a maximal cycle length of m^k. However, the method has also some disadvantages. Consider for example the special case $k = 2$. Now, if we take three consecutive numbers U_{i-2}, U_{i-1} and U_i, it will never happen that $U_{i-2} < U_i < U_{i-1}$ or $U_{i-1} < U_i < U_{i-2}$. (For three independent, uniformly (0,1) distributed random variables both of these orderings have probability 1/6).

4.1.4 Tausworthe Generators

The recursions in the linear congruential method and in the additive congruential are special cases of the recursion

$$Z_i = (\sum_{j=1}^{k} a_j Z_{i-j} + c) \bmod m, \qquad i = 1, 2, 3, \ldots.$$

A special situation is obtained if we take $m = 2$ and all $a_j \in \{0, 1\}$. In that case the generator produces a sequence of bits. Such a generator is often called a *Tausworthe generator*, or a *shift register generator*. The sequence $Z_0, Z_1, Z_2, \ldots$, is now transformed into a sequence $U_0, U_1, U_2, \ldots$, of uniform random numbers in the following way: Choose an integer $l \leq k$ and put

$$U_n = \sum_{j=0}^{l-1} Z_{nl+j} 2^{-(j+1)}.$$

In other words, the numbers U_n are obtained by splitting up the sequence $Z_0, Z_1, Z_2, \ldots$ into consecutive blocks of length l and then interpreting each block as the digit expansion in base 2 of a number in [0,1).

In practice, often only two of the a_j's are chosen equal to one, so we get

$$Z_i = (Z_{i-h} + Z_{i-k}) \bmod 2.$$

Example 4.3 *If we choose* $h = 3$ *and* $k = 4$ *and start with the initial values 1,1,0,1, then we get the following sequence of bits* 1, 1, 0, 1, 0, 1,1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,1, 0, 1, 1, 1, 1, ... *The sequence is periodic with period* $2^k - 1 = 15$. *If we take* $l = 4$, *this leads to the random numbers* 13/16, 7/16, 8/16, 9/16, 10/16, 15/16, 1/16, 3/16, ...

4.2 Tests of Random Number Generators

The main properties that a random number generator should have are uniformity and independence. In this section we will describe two tests, the Kolmogorov-Smirnov test and the chi-square test, to compare the distribution of the set of generated numbers with a uniform distribution. Furthermore, we will describe a number of tests that are able to check whether or not the set of generated numbers satisfies independence.

4.2.1 The Kolmogorov-Smirnov test

This test compares the empirical distribution $F_N(x)$ of a set of N generated random numbers with the distribution function $F(x) = x, 0 \leq x \leq 1$, of a uniformly distributed random variable. Here, $F_N(x)$ is defined as the number of observations smaller than or equal to x divided by N, the total number of observations. Let α be the significance level of the test, i.e. α is the probability of rejecting the null hypothesis that the numbers are uniformly distributed on the interval (0, 1) given that the null hypothesis is true. Under the null hypothesis $F_N(x)$ will tend to $F(x)$ as N tends to infinity. The Kolmogorov-Smirnov test is based on

$$D = \max | F(x) - F_N(x) | ,$$

the maximal absolute difference between $F_N(x)$ and $F(x)$ over the range of the random variable. Now, if the value of D is greater than some critical value D_α the null hypothesis is rejected. If $D \leq D_\alpha$, we conclude that no difference has been detected between the empirical distribution of the generated numbers and the uniform distribution. The critical value D_α for the specified significance level α of the test and the given sample size N can be found in a table on the Kolmogorov-Smirnov test.

4.2.2 The chi-square test

In the chi-square test, we divide the interval (0, 1) in n subintervals of equal length. The test uses the statistic

$$Y = \sum_{i=1}^{n} \frac{(Y_i - E(Y_i))^2}{E(Y_i)},$$

where Y_i is the number of realizations in the i-th subinterval. Clearly, if N is the total number of observations, we have $E(Y_i) = N/n$. Under the null hypothesis of uniformity of the realizations, it can be shown that the distribution of Y is approximately the chi-square distribution with $n - 1$ degrees of freedom. So, we compare the value of our statistic Y with the α-percentile of the chi-square distribution with $n - 1$ degrees of freedom to conclude whether or not we reject the null hypothesis. Here, α is again the significance level of the test. It is recommended to choose n and N such that $N \geq 50$ and $E(Y_i) \geq 5$.

4.2.3 The serial test

The serial test is a 2-dimensional version of the uniformity test of the previous subsection to test the independence between successive observations. Therefore, we look at N successive tuples $(U_1, U_2), (U_3, U_4), \ldots, (U_{2N-1}, U_{2N})$ of our observations and we count how many observations fall into the N^2 different subsquares of the unit square. We then apply a chi-square test to these data. Of course, we can also formulate a higher-dimension version of this test.

4.2.4 The permutation test

In the permutation test we look at N successive k-tuples of realizations $(U_0, \ldots, U_{k-1})$, $(U_k, \ldots, U_{2k-1})$, $\ldots$, $(U_{(N-1)k}, \ldots, U_{Nk-1})$. Among the elements in a k-tuple there are $k!$ possible orderings and these orderings are all equally likely. Hence, we can determine the frequencies of the different orderings among the N different k-tupels and apply a chi-squared test to these data.

4.2.5 The run test

In the run test we divide the realizations in blocks, where each block consists of a sequence of increasing numbers followed by a number which is smaller than its predecessor. For example, if the realizations are 1,3,8,6,2,0,7,9,5, then we divide them in the blocks (1,3,8,6), (2,0) and (7,9,5). A block consisting of $k+1$ numbers is called a *run-up of length* k. Under the null hypothesis that we are dealing with independent, uniformly distributed random variables the probability of having a run-up of length k equals $1/k! - 1/(k+1)!$. In a practical implementation of the run test we now observe a large number N of blocks. Furthermore, we choose an integer h and count the number of runs-up of length $0, 1, 2, \ldots, h-1$ and $\geq h$. Then we apply a chi-square test to these data. Of course, we can do a similar test for runs-down.

4.2.6 The gap test

Let J be some fixed subinterval of (0, 1). If we have that $U_{n+j} \notin J$ for $0 \leq j \leq k$ and both $U_{n-1} \in J$ and $U_{n+k+1} \in J$, we say that we have a *gap of length* k. Under the null hypothesis that the random numbers U_n are independent and uniformly distributed on (0,1), we have that the gap lengths are geometrically distributed with parameter p, where p is the length of interval J (i.e. P(gap of length k) = $p(1-p)^k$). In a practical implementation of the gap test we observe again a large number N of gaps. Furthermore, we choose an integer h and count the number of gaps of length $0, 1, 2, \ldots, h-1$ and $\geq h$. Then we apply a chi-square test to these data.

4.2.7 The serial correlation test

In the serial correlation test we calculate the serial correlation coefficient

$$R = \sum_{j=1}^{N} (U_j - \bar{U})(U_{j+1} - \bar{U}) / \sum_{j=1}^{N} (U_j - \bar{U})^2,$$

where $\bar{U} = \sum_{j=1}^{N} U_j / N$ and U_{N+1} should be replaced by U_1. If the U_j's are really independent the serial correlation coefficient should be close to zero. Hence we reject the null hypothesis that the U_j's are independent if R is too large. The exact distribution of R is unknown, but for large values of N we have that, if the U_j's are independent, then $P(-2/\sqrt{N} \leq R \leq 2/\sqrt{N}) \approx 0.95$. Hence, we reject the hypothesis of independence at the 5% level if $R \notin (-2/\sqrt{N}, 2/\sqrt{N})$.

5 Generating Random Variables

In this section we discuss a number of approaches for generating random variables. First, we discuss the generation of discrete random variables. After that, we will introduce several methods to generate continuous random variables.

5.1 Generating Discrete Random Variables

Let X be a discrete random variable with arbitrary probabilities

$$P(X = x_j) = p_j, \qquad j = 0, 1, 2, \ldots, \qquad \sum_{j=0}^{\infty} p_j = 1.$$

To generate a realization of X, we generate a random number U and set

$$X = \begin{cases} x_0 & \text{if} \quad U \in [0, p_0), \\ x_1 & \text{if} \quad U \in [p_0, p_0 + p_1), \\ \vdots & \\ x_j & \text{if} \quad U \in [\sum_{i=0}^{j-1} p_i, \sum_{i=0}^{j} p_i), \\ \vdots & \end{cases}$$

A nice feature of this method is that we always need exactly one uniformly distributed variable to generate one realization of the desired random variable. Other methods that we will discuss require often more than one uniformly distributed variables to generate one realization of the desired random variable. This observation can be important when you want to use simulation for the comparison of different systems and you want to synchronize the input random streams.

Although this method can be used for the generation of arbitrary discrete random variables, it is not always the most efficient method. Hence we shall give some alternative methods for some special distributions.

5.1.1 The geometric distribution

A random variable X has a geometric distribution with parameter p if

$$P(X = k) = p(1 - p)^{k-1}, \quad k = 1, 2, 3, \ldots$$

The geometric distribution can be interpreted as the distribution of the waiting time until the first head in a sequence of independent coin tossing experiments, where p equals the probability of throwing a head. Hence, if $U_1, U_2, \ldots$ are independent, identically distributed uniform (0,1) random variables and X is the index of the first U_i for which $U_i \leq p$, then X is geometrically distributed with parameter p.

Another way to generate a geometric random variable is by using the fact that if U is uniform (0,1), then

$$\left\lceil \frac{\ln (U)}{\ln (1 - p)} \right\rceil$$

is geometric with parameter p.

5.1.2 The binomial distribution

A random variable X has a binomial distribution with parameters n and p if

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, 2, \ldots, n.$$

The binomial distribution can be interpreted as the distribution of the number of heads in a sequence of n independent coin tossing experiments, where p equals the probability of throwing a head. Hence, if $U_1, \ldots, U_n$ are independent, identically distributed uniform (0,1) random variables, then

$$X = \sum_{i=1}^{n} 1_{[U_i \leq p]}$$

is binomial distributed with parameters n and p. An alternative generation of a binomial distributed random variable can be obtained using one of the following two lemma's.

Lemma 5.1 *Let $G_1, G_2, \ldots$ be independent, identically distributed geometric random variables with parameter p. If X is the smallest integer such that*

$$\sum_{i=1}^{X+1} G_i > n,$$

then X is binomial distributed with parameters n and p.

Lemma 5.2 *Let $E_1, E_2, \ldots$ be independent, identically distributed exponential random variables with parameter 1. If X is the smallest integer such that*

$$\sum_{i=1}^{X+1} \frac{E_i}{n-i+1} > -\ln(1-p),$$

then X is binomial distributed with parameters n and p.

5.1.3 The Poisson distribution

A random variable X has a Poisson distribution with parameter λ if

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}, \quad k = 0, 1, 2, \ldots.$$

An alternative generation of a Poisson distributed random variable can be obtained using one of the following two lemma's.

Lemma 5.3 *Let $E_1, E_2, \ldots$ be independent, identically distributed exponential random variables with parameter 1. If X is the smallest integer such that*

$$\sum_{i=1}^{X+1} E_i > \lambda,$$

then X is Poisson distributed with parameter λ.

Lemma 5.4 *Let $U_1, U_2, \ldots$ be independent, identically distributed uniform (0,1) random variables. If X is the smallest integer such that*

$$\prod_{i=1}^{X+1} U_i < e^{-\lambda},$$

then X is Poisson distributed with parameter λ.

5.2 Generating Continuous Random Variables

We describe four general methods to generate continuous random variables: the inverse transform method, the acceptance-rejection method, the composition method and the convolution method. After that, we pay special attention to the generation of normally distributed random variables.

5.2.1 Inverse Transform Method

The inverse transform method is applicable when the distribution function F of a random variable is continuous and strictly increasing on the domain of the random variable. In that case, the inverse function F^{-1} exists and it is easily seen that, if U in uniformly distributed on $(0, 1)$, then $X = F^{-1}(U)$ has distribution function F:

$$P(X \leq x) = P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x).$$

Example 5.5

- $X = a + (b - a)U$

  is uniformly distributed on the interval (a, b).

- $X = -\frac{1}{\lambda}\, ln\, (1 - U)$

  is exponentially distributed with parameter λ.

- $X = \frac{1}{\lambda}(-\, ln\, (1 - U)^{1/a}$

  is Weibull distributed with parameters a and λ.

If F is not continuous or not strictly increasing, then instead of working with the inverse function we have to introduce the generalized inverse function

$$F^{-1}(u) := \min\{x : F(x) \geq u\}.$$

If we do so, then the general method described in section 5.1 for generating discrete random variables is in fact a special case of the inverse transform method.

Unfortunately, for some often used distribution functions like the normal distribution and the Erlang distribution we do not have a closed form expression for F^{-1}.

5.2.2 Acceptance-Rejection Method

Suppose that we want to generate a random variable X with distribution function F and probability density function f. The acceptance-rejection method requires a function g which majorizes the density f, i.e. $g(x) \geq f(x)$ for all x. Clearly, $g(x)$ will not be a probability density function because $c := \int g(x)dx > 1$. However, if $c < \infty$, then $h(x) = g(x)/c$ is a density. The method applies if we are able to generate easily a random variable Y with density h. The method consists of three steps. First, we generate Y having density h. Next, we generate a uniformly distributed random variable U on the interval $(0, 1)$. Finally, we set $X = Y$ if $U \leq f(Y)/g(Y)$ or, if not, we go back to the first step.

Theorem 5.6 *The random variable X generated by the acceptance-rejection method has probability density function f. The number of iterations of the algorithm that are needed is a geometric random variable with mean c.*

Proof: The probability that a single iteration produces an accepted value which is smaller than x equals

$$P(Y \leq x, Y \text{ is accepted}) = \int_{-\infty}^{x} \frac{f(y)}{g(y)} h(y)dy = \frac{1}{c} \int_{-\infty}^{x} f(y)dy$$

and hence each iteration is accepted with probability $1/c$. As each iteration is independent, we see that the number of iterations needed is geometric with mean c. Furthermore,

$$P(X \leq x) = \sum_{n=1}^{\infty} (1 - 1/c)^{n-1} 1/c \int_{-\infty}^{x} f(y)dy = \int_{-\infty}^{x} f(y)dy.$$

5.2.3 Composition Method

The composition method applies when the distribution function F from which we want to sample can be expressed as a convex combination of other distribution functions $F_1, F_2, \ldots$, i.e.

$$F(x) = \sum_{j=1}^{\infty} p_j F_j(x)$$

with $p_j \geq 0$ for all j and $\sum_{j=1}^{\infty} p_j = 1$. The method is useful when it is more easy to generate random variables with distribution function F_j then to generate a random variable with distribution function F. The method consists of two steps. First, we generate a discrete random variable J, such that $P(J = j) = p_j$. Next, given $J = j$, we generate a random variable X with distribution function F_j. It is easily seen that the random variable X has the desired distribution function F. The method can be used, for example, to generate a hyperexponentially distributed random variable.

5.2.4 Convolution Method

For some important distributions, the desired random variable X can be expressed as the sum of other independent random variables, i.e. $X = Y_1 + \cdots + Y_n$, which can be generated more easily than the random variable X itself. The convolution method simply says that you first generate the random variables Y_j and then add them to obtain a realization of the random variable X. We can apply this method, for example, to obtain Erlang distributed random variable. However, when n, the number of phases of the Erlang distribution, is large this may be not the most efficient way to generate an Erlang distributed random variable.

5.2.5 Generating Normal Random Variables

In this section we will show three methods to generate standard normal random variables. The first method is an approximative method using the central limit theorem. The second and third method are exact methods. The second method uses the acceptance-rejection method. The third method is a method developed by Box and Muller which generates two independent standard normal random variables by a transformation of two independent uniform random variables.

Method 1: Using the Central Limit Theorem

An easy, but not very good, approximative method of generating standard normally distributed random variables makes use of the central limit theorem. Recall that the central limit theorem states that for a sequence $Y_1, Y_2, \ldots$ of independent, identically distributed random variables with mean μ and variance σ^2,

$$\frac{\sum_{i=1}^{n} Y_i - n\mu}{\sigma\sqrt{n}} \xrightarrow{d} N(0, 1),$$

where $\xrightarrow{d}$ means convergence in distribution and $N(0, 1)$ is a standard normal random variable. If we assume that $Y_i = U_i$, i.e. Y_i is uniformly distributed on (0,1), then we have $\mu = 1/2$ and $\sigma^2 = 1/12$. Hence we can take e.g. $\sum_{i=1}^{12} U_i - 6$ as an approximative standard normal random variable.

Method 2: Using the Acceptance-Rejection Method

The probability density function of the absolute value of a standard normal random value equals

$$f(x) = \frac{2}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}, \qquad 0 < x < \infty.$$

Now, the function

$$g(x) = \sqrt{2e/\pi}\, e^{-x}$$

majorizes the probability density function f. (Check!) We conclude that we can generate a standard normal random variable in the following way: Generate an exponential random variable Y with parameter 1 and a uniform random variable U on the interval $(0, 1)$. Accept Y if

$U \leq e^{-(Y-1)^2/2}$, otherwise repeat this procedure. Finally, set $X = Y$ or $X = -Y$, both with probability 1/2.

Method 3: Using the Box-Muller Method

Theorem 5.7 *If U_1 and U_2 are two independent, uniformly* $(0, 1)$ *distributed random variables, then*

$$\begin{aligned} X_1 &= \sqrt{-2\ ln\ U_1}\ cos\ (2\pi U_2) \\ X_2 &= \sqrt{-2\ ln\ U_1}\ sin\ (2\pi U_2) \end{aligned}$$

are two independent, standard normally distributed random variables.

Proof: The joint density function of two independent, standard normal random variables (X_1, X_2) equals

$$f(x_1, x_2) = \frac{1}{2\pi} e^{-\frac{1}{2}(x_1^2 + x_2^2)}, \qquad 0 < x_1, x_2 < \infty.$$

Introducing polar coordinates, i.e. $X_1 = R \cos \Phi$ and $X_2 = R \sin \Phi$ we get for the joint density function of R and Φ

$$g(r, \phi) = \frac{1}{2\pi} r e^{-\frac{1}{2}r^2}, \qquad 0 < \phi < 2\pi, \quad 0 < r < \infty.$$

We conclude that R and Φ are independent. Furthermore, it is easy to check that R has the same density as $\sqrt{-2 \ln U_1}$ and Φ has the same density as $2\pi U_2$, where U_1 and U_2 are uniformly $(0, 1)$ distributed. Hence the theorem follows.

5.3 Generating Poisson Processes

In practice, we often want to simulate a homogeneous or non-homogeneous Poisson process. The generation of a homogeneous Poisson process can of course be done by using the fact that this process has independent, exponentially distributed interarrival times. For the simulation of a non-homogeneous Poisson process we can use a thinning approach. Suppose that we want to simulate a non-homogeneous Poisson process with arrival rate $\lambda(t)$ at time t until time T. Assume that $\lambda(t) < \lambda$ for all $t \in [0, T]$. We can simulate this process in the following way: we generate a Poisson process with rate λ and accept a possible arrival at time t with probability $\lambda(t)/\lambda$.

Another popular arrival process is a so-called *Markov modulated Poisson process*. For this process, the rate at which, say, customers arrive is modulated by an underlying Markov process. If the underlying Markov process is in state i, then the arrival rate of the Poisson process equals λ_i. Find a way to generate a Markov modulated Poisson process.