Chapter 2
Generating Random Numbers with Specified Distributions

Simulation and valuation of finance instruments require numbers with specified distributions. For example, in Sect. 1.6 we have used numbers Z drawn from a standard normal distribution, $Z \sim \mathcal{N}(0, 1)$. If possible the numbers should be random. But the generation of "random numbers" by digital computers, after all, is done in a deterministic and entirely predictable way. If this point is to be stressed, one uses the term *pseudo-random*.[1]

Computer-generated random numbers mimic the properties of true random numbers as much as possible. This is discussed for uniformly distributed random numbers in Sect. 2.1. Suitable transformations or rejection methods generate samples from other distributions, in particular, normally distributed numbers (Sects. 2.2 and 2.3). Section 2.3 includes the vector case, where normally distributed numbers are calculated with prescribed correlation.

Another approach is to dispense with randomness and to generate *quasi-random numbers*, which aim at avoiding one disadvantage of random numbers, namely, the potential lack of equidistributedness. The resulting *low-discrepancy* numbers will be discussed in Sect. 2.5. These numbers are used for the deterministic Monte Carlo integration (Sect. 2.4).

Definition 2.1 (Sample from a Distribution) A sequence of numbers is called a *sample from F* if the numbers are independent realizations of a random variable with distribution function F.
If F is the uniform distribution over the interval $[0, 1]$, then we call the samples from F *uniform deviates (variates)*, notation $\sim \mathcal{U}[0, 1]$. If F is the standard normal distribution then we call the samples from F *standard normal deviates (variates)*; as notation we use $\sim \mathcal{N}(0, 1)$. The basis of random-number generation is to draw uniform deviates.

[1]Since in our context the predictable origin is clear we omit the modifier "pseudo," and hereafter use the term "random number." Similarly we talk about randomness of these numbers when we mean apparent randomness.

© Springer-Verlag London Ltd. 2017

R.U. Seydel, *Tools for Computational Finance*, Universitext,
DOI 10.1007/978-1-4471-7338-0_2

2.1 Uniform Deviates

A standard approach to calculate uniform deviates is provided by linear congruential generators. We concentrate on algorithms that are easy to implement and ready for experiments.

2.1.1 Linear Congruential Generators

Choose integers M, a, b, with $a, b < M$, $a \neq 0$. For an integer N_0 a sequence of integers N_i is defined by

Algorithm 2.2 (Linear Congruential Generator)

```
Choose $N_0$.
For $i = 1, 2, \ldots$ calculate
    $N_i = (aN_{i-1} + b) \bmod M$.   (2.1)
```

The *modulo* congruence $N = Y \bmod M$ between two numbers N and Y is an equivalence relation [147]. The initial integer N_0 is called the *seed*. Numbers $U_i \in [0, 1)$ are defined by

$$U_i = N_i/M\,, \tag{2.2}$$

and will be taken as uniform deviates. Whether the numbers U_i or N_i are suitable will depend on the choice of M, a, b and will be discussed next.

Properties 2.3 (Periodicity)

(a) $N_i \in \{0, 1, \ldots, M-1\}$
(b) The N_i are periodic with period $\leq M$.
(Because there are not $M + 1$ different N_i. So two in $\{N_0, \ldots, N_M\}$ must be equal, $N_i = N_{i+p}$ with $p \leq M$.)

Obviously, some peculiarities must be excluded. For example, $N = 0$ must be ruled out in case $b = 0$, because otherwise $N_i = 0$ would repeat. In case $a = 1$ the generator settles down to $N_n = (N_0 + nb) \bmod M$. This sequence is predictable too easily. Various other properties and requirements are discussed in the literature, in particular in [226]. In case the period is M, the numbers U_i are distributed "evenly" when exactly M numbers are needed. Then each grid point on a mesh on [0,1] with mesh size $\frac{1}{M}$ is occupied once.

After these observations we start searching for good choices of M, a, b. There are numerous possible choices with bad properties. For serious computations we recommend to rely on suggestions of the literature. Press et al. [306] presents a table of "quick and dirty" generators, for example, $M = 244{,}944$, $a = 1597$,

$b = 51{,}749$. Criteria are needed to decide which of the many possible generators are recommendable.

2.1.2 Quality of Generators

What are good random numbers? A practical answer is the requirement that the numbers should meet "all" aims, or rather pass as many tests as possible. The requirements on good number generators can roughly be divided into three groups.

The first requirement is that of a *large period*. In view of Property 2.3 the number M must be as large as possible, because a small set of numbers makes the outcome easier to predict—a contrast to randomness. This leads to select M close to the largest integer machine number. But a period p close to M is only achieved if a and b are chosen properly. Criteria for relations among M, p, a, b have been derived by number-theoretic arguments. This is outlined in [226, 317]. For 32-bit computers, a common choice has been $M = 2^{31} - 1$, $a = 16807$, $b = 0$.

A second group of requirements are *statistical tests* that check whether the numbers are distributed as intended. The simplest of such tests evaluates the sample mean $\hat{\mu}$ and the sample variance $\hat{s}^2$ (B.11) of the calculated random variates, and compares to the desired values of μ and σ^2. (Recall $\mu = 1/2$ and $\sigma^2 = 1/12$ for the uniform distribution.) Another simple test is to check correlations. For example, it would not be desirable if small numbers are likely to be followed by small numbers.

A slightly more involved test checks how well the probability distribution is approximated. This works for general distributions ($\longrightarrow$ Exercise 2.1). Here we briefly summarize an approach for uniform deviates. Calculate j samples from a random number generator, and investigate how the samples distribute on the unit interval. To this end, divide the unit interval into subintervals of equal length ΔU, and denote by j_k the number of samples that fall into the kth subinterval

$$k\Delta U \le U < (k+1)\Delta U\,.$$

Then j_k/j should be close the desired probability, which for this setup is ΔU. Hence a plot of the quotients

$$\frac{j_k}{j\Delta U} \quad \text{for all } k$$

against $k\Delta U$ should be a good approximation of $\mathbf{1}_{[0,1]}$, the density of the uniform distribution. This procedure is just the simplest test; for more ambitious tests, consult [226].

The third group of tests is to check how well the random numbers distribute in higher-dimensional spaces. This issue of the *lattice structure* is discussed next. We derive a priori analytical results on *where* the random numbers produced by Algorithm 2.2 are distributed.

2.1.3 Random Vectors and Lattice Structure

Random numbers N_i can be arranged in m-tuples $(N_i, N_{i+1}, \ldots, N_{i+m-1})$ for $i \geq 1$. Then the tuples or the corresponding points $(U_i, \ldots, U_{i+m-1}) \in [0,1)^m$ are analyzed with respect to correlation and distribution. The sequences defined by the generator of Algorithm 2.2 lie on $(m-1)$-dimensional hyperplanes. This statement is trivial since it holds for the M parallel planes through $U = i/M$, $i = 0, \ldots, M-1$ (any of the m components). But if all points fall on only a *small* number of parallel hyperplanes (with large empty gaps in between), then the generator would be impractical in many applications. Next we analyze the generator whether such unfavorable planes exist, restricting ourselves to the case $m = 2$.

For $m = 2$ the hyperplanes in (U_{i-1}, U_i)-space are straight lines, and are defined by $z_0 U_{i-1} + z_1 U_i = \lambda$, with parameters z_0, z_1, λ. The modulus operation (2.1) can be written

$$\begin{aligned} N_i &= (aN_{i-1} + b) \bmod M \\ &= aN_{i-1} + b - kM \quad \text{for } kM \leq aN_{i-1} + b < (k+1)M, \end{aligned}$$

k an integer, $k = k(i)$. A side calculation for arbitrary z_0, z_1 shows

$$\begin{aligned} z_0 N_{i-1} + z_1 N_i &= z_0 N_{i-1} + z_1 (aN_{i-1} + b - kM) \\ &= N_{i-1}(z_0 + az_1) + z_1 b - z_1 kM \\ &= M \cdot \underbrace{\{N_{i-1} \frac{z_0 + az_1}{M} - z_1 k\}}_{=:c} + z_1 b\,. \end{aligned}$$

We divide by M and obtain the equation of a straight line in the (U_{i-1}, U_i)-plane, namely,

$$z_0 U_{i-1} + z_1 U_i = c + z_1 b M^{-1}\,. \tag{2.3}$$

The points calculated by Algorithm 2.2 lie on these straight lines. To eliminate the seed we take $i > 1$. For each tuple (z_0, z_1), the Eq. (2.3) defines a family of parallel straight lines, one for each number out of the finite set of c's. The question is whether there exists a tuple (z_0, z_1) such that only few of the straight lines cut the square $[0,1)^2$. In this case wide areas of the square would be free of random points, which violates the requirement of a "uniform" distribution of the points. The minimum number of parallel straight lines (hyperplanes) cutting the square, or equivalently the maximum distance between them, characterizes the worst case and serves as measure of the equidistributedness. Now we analyze the number of straight lines, searching for the worst case.

For analyzing the number of planes, the cardinality of the c matters. To find the worst case, restrict to integers (z_0, z_1) satisfying

$$z_0 + az_1 = 0 \bmod M\,. \tag{2.4}$$

Then the parameter c is integer. By solving (2.3) for $c = z_0 U_{i-1} + z_1 U_i - z_1 b M^{-1}$ and applying $0 \le U < 1$ we obtain the maximal interval I_c such that for each integer $c \in I_c$ its straight line cuts or touches the square $[0, 1)^2$. Count how many such c's exist, and there is the information we need. For some constellations of a, M, z_0 and z_1 it may be possible that the points (U_{i-1}, U_i) lie on very few of these straight lines!

Example 2.4 (Academic Generator) We discuss the generator

$$N_i = 2N_{i-1} \bmod 11$$

that is, the parameters are $a = 2,\ b = 0,\ M = 11$. The choice $z_0 = -2,\ z_1 = 1$ is one tuple satisfying (2.4), and the resulting family (2.3) of straight lines

$$-2U_{i-1} + U_i = c$$

in the (U_{i-1}, U_i)-plane is to be discussed. For $U \in [0, 1)$ the inequality $-2 < c < 1$ results. In view of (2.4) c is integer and so only the two integers $c = -1$ and $c = 0$ remain. The two corresponding straight lines cut the interior of $[0, 1)^2$. As Fig. 2.1 illustrates, the points generated by the algorithm form a lattice. All points on the lattice lie on these two straight lines. The figure lets us discover also other parallel straight lines such that all points are caught (for other tuples z_0, z_1). The practical question is: What is the largest gap? ($\longrightarrow$ Exercise 2.2)

Example 2.5 $N_i = (1229N_{i-1} + 1) \bmod 2048$

The requirement of Eq. (2.4)

$$\frac{z_0 + 1229z_1}{2048} \text{ integer}$$

is satisfied by $z_0 = -1,\ z_1 = 5$, because

$$-1 + 1229 \cdot 5 = 6144 = 3 \cdot 2048\,.$$

For c from (2.3) and $U_i \in [0, 1)$ we have

$$-1 - \frac{5}{2048} < c < 5 - \frac{5}{2048}\,.$$

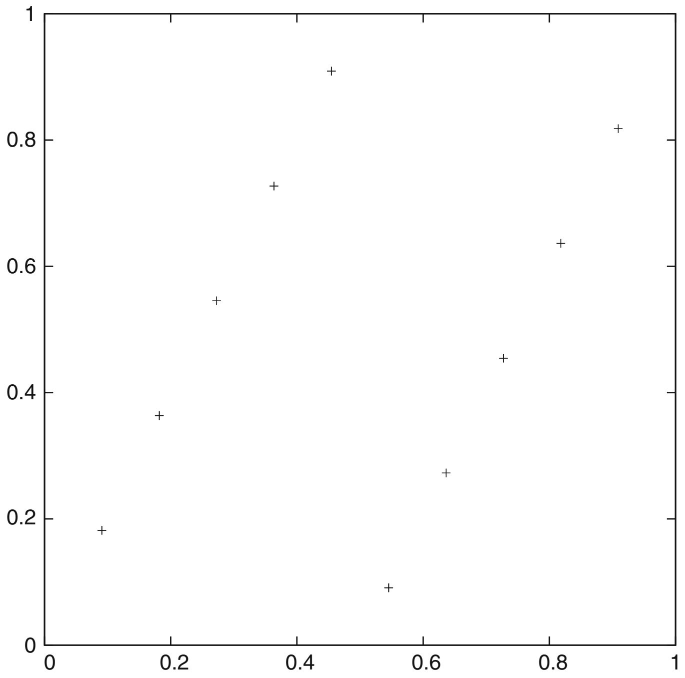

Fig. 2.1 The points (U_{i-1}, U_i) of Example 2.4

Hence $c \in \{-1, 0, 1, 2, 3, 4\}$, and all points (U_{i-1}, U_i) in $[0, 1)^2$ lie on only six straight lines, see Fig. 2.2. On the "lowest" straight line ($c = -1$) there is only one point. The distance between straight lines measured along the vertical U_i–axis is $\frac{1}{z_1} = \frac{1}{5}$. Obviously, the (U_{i-1}, U_i)-points are by far not equidistributed on the square, although the positions U_i appear uniformly distributed on the line.[2]

Higher-dimensional vectors ($m > 2$) are analyzed analogously. The generator called RANDU

$$N_i = aN_{i-1} \bmod M\,, \quad \text{with } a = 2^{16} + 3,\; M = 2^{31}$$

may serve as example. For $m = 2$ experiments show that the points (U_{i-1}, U_i) are nicely equidistributed. But equidistribution for $m = 2$ does not imply equidistribution for larger m. Testing RANDU for $m = 3$ reveals a severe defect: Its random points in the cube $[0, 1)^3$ fall on only 15 planes ($\longrightarrow$ Exercise 2.3 and Topic 14 in the *Topics fCF*).

In Example 2.4 we asked what the maximum gap between the parallel straight lines is. In other words, we have searched for stripes of maximum size in which

[2]The term "equidistributed" will be quantified in Sect. 2.5.1.

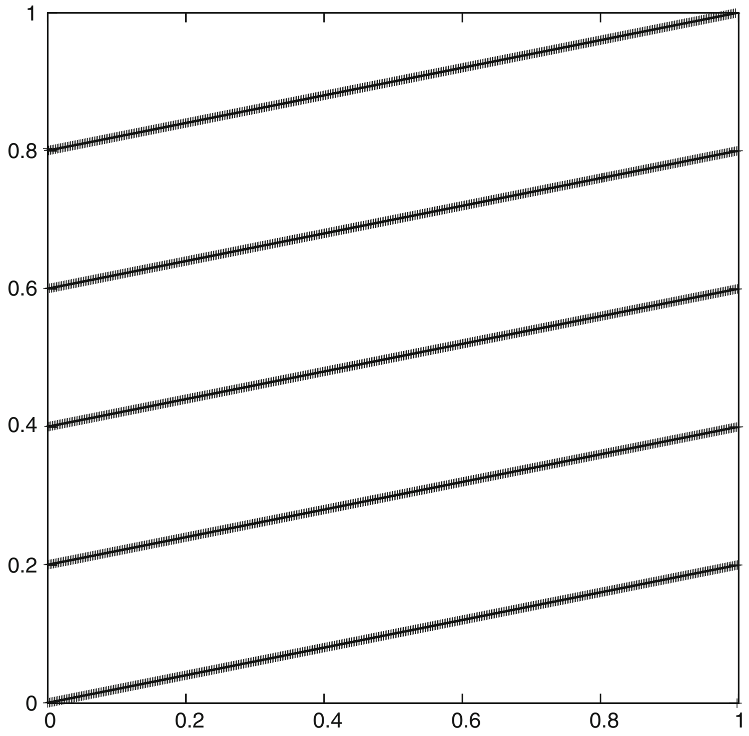

Fig. 2.2 The points (U_{i-1}, U_i) of Example 2.5

no point (U_{i-1}, U_i) falls. Alternatively one can directly analyze the lattice formed by consecutive points. For illustration consider again Fig. 2.1. We follow the points starting with $(\frac{1}{11}, \frac{2}{11})$. By vectorwise adding an appropriate multiple of $(1, a) = (1, 2)$ the next two points are obtained. Proceeding in this way one has to take care that upon leaving the unit square each component with value ≥ 1 must be reduced to $[0, 1)$ to observe $\bmod M$. The reader may verify this with Example 2.4 and numerate the points of the lattice in Fig. 2.1 in the correct sequence. In this way the lattice can be defined. This process of defining the lattice can be generalized to higher dimensions $m > 2$. ($\longrightarrow$ Exercise 2.4) One aims at a good distribution of the points $(U_i, \ldots, U_{i+m-1})$ for as many m are possible.

A disadvantage of the linear congruential generators of Algorithm 2.2 is the boundedness of the period by M and hence by the word length of the computer. The situation can be improved by *shuffling* the random numbers in a random way. For practical purposes, the period gets close enough to infinity. (The reader may test this on Example 2.5.) For practical advice we refer to [306].

2.1.4 Fibonacci Generators

The original Fibonacci recursion motivates trying the formula

$$N_{i+1} := (N_i + N_{i-1}) \bmod M \, .$$

It turns out that this first attempt of a three-term recursion is not suitable for generating random numbers ($\longrightarrow$ Exercise 2.5). The modified approach

$$N_{i+1} := (N_{i-\nu} - N_{i-\mu}) \bmod M \tag{2.5}$$

for suitable integers ν, μ is called *lagged Fibonacci* generator. For many choices of ν, μ the approach (2.5) leads to acceptable generators. Kahaner et al. [210] recommends

Example 2.6 (Lagged Fibonacci Generator)

$$U_i : = U_{i-17} - U_{i-5} \, ,$$
$$\text{in case } U_i < 0 \text{ set } U_i := U_i + 1.0 \, .$$

The recursion of Example 2.6 immediately produces floating-point numbers $U_i \in [0, 1)$. This generator requires a prologue in which 17 initial U's are generated by means of another method. The core of the algorithm is

Algorithm 2.7 (Loop of a Fibonacci Generator)

```
Repeat:
    $\zeta = U(i) - U(j)$,
    if $(\zeta < 0)$, set $\zeta = \zeta + 1$,
    $U(i) = \zeta$,
    $i = i - 1$,
    $j = j - 1$,
    if $i = 0$, set $i = 17$,
    if $j = 0$, set $j = 17$.
```

Initialization: Set $i = 17$, $j = 5$, and calculate $U_1, \ldots, U_{17}$ with a congruential generator, for instance with $M = 714{,}025$, $a = 1366$, $b = 150{,}889$. Set the seed N_0 equal to your favorite dream number, possibly inspired by the system clock of your computer.

Figure 2.3 depicts 10, 000 random points calculated by means of Algorithm 2.7. Visual inspection suggests that the points are not arranged in some apparent structure. The points appear to be sufficiently random. But the generator provided by Example 2.6 is not sophisticated enough for ambitious applications; its pseudo-random numbers are somewhat correlated.

Section 2.1 has introduced some basic aspects of generating uniformly distributed random numbers. Professional algorithms also apply bit operations in

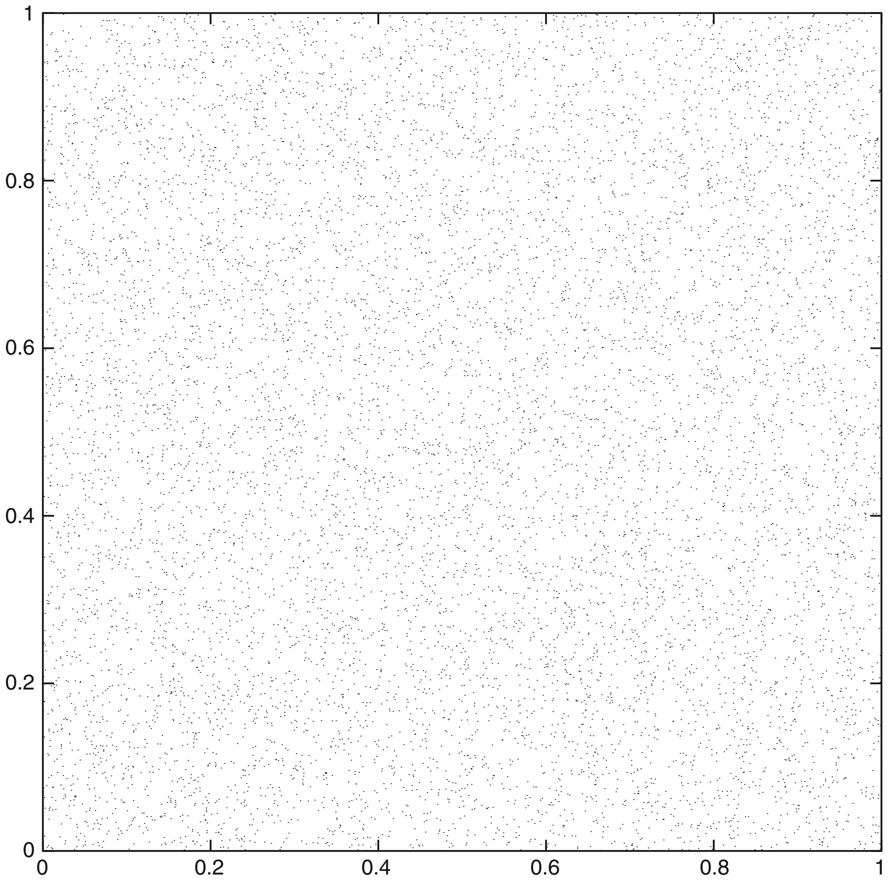

Fig. 2.3 Ten thousand (pseudo-)random points (U_{i-1}, U_i), calculated with Algorithm 2.7

the computer. A generator of uniform deviates that can be highly recommended is a *Mersenne twister* [264]. Its period is truly remarkable, and the points $(U_i, \ldots, U_{i+m-1})$ are well distributed until high values of the dimension m.

2.2 Extending to Random Variables from Other Distributions

Frequently, normal variates are needed. Their generation is based on uniform deviates. The simplest strategy is to calculate

$$X := \sum_{i=1}^{12} U_i - 6, \quad \text{for } U_i \sim \mathcal{U}[0,1]\,.$$

X has expectation 0 and variance 1. The central limit theorem ($\longrightarrow$ Appendix B) assures that X is approximately distributed normally ($\longrightarrow$ Exercise 2.6). But this crude attempt is not satisfying. Better methods calculate nonuniformly distributed random variables, for example, by a suitable transformation out of a uniformly distributed random variable [103]. But the most obvious approach inverts the distribution function.

2.2.1 Inversion

The following theorem is the basis for inversion methods.

Theorem 2.8 (Inversion) *Suppose $U \sim \mathcal{U}[0,1]$ and F be a continuous strictly increasing distribution function. Then $F^{-1}(U)$ is a sample from F.*

Proof Let P denote the underlying probability.
$U \sim \mathcal{U}[0,1]$ means $\quad \mathsf{P}(U \le \xi) = \xi$ for $0 \le \xi \le 1$.
Consequently

$$\mathsf{P}(F^{-1}(U) \le x) = \mathsf{P}(U \le F(x)) = F(x)\,.$$

Application
Following Theorem 2.8, the inversion method[3] generates uniform deviates $u \sim \mathcal{U}[0,1]$ and sets $x = F^{-1}(u)$ ($\longrightarrow$ Exercises 2.7, 2.8, 2.9). There are some examples where the inverse is available analytically. For example, the distribution of the exponential distribution with parameter λ (below in Example 2.10) is $F(x) = 1 - \mathrm{e}^{-\lambda x}$, and its inverse is $F^{-1}(u) = -\frac{1}{\lambda}\log(1-u)$. To judge the inversion method we consider the normal distribution as the most important example. Neither for its distribution function F nor for its inverse F^{-1} there is a closed-form expression ($\longrightarrow$ Exercise 1.5). So numerical methods are used. We discuss two approaches.

Numerical inversion means to calculate iteratively a solution x of the equation $F(x) = u$ for prescribed u. In particular for the normal distribution, this iteration requires tricky termination criteria, in particular when x is large. Then we are in the situation $u \approx 1$, where tiny changes in u lead to large changes in x (Fig. 2.4). An approximation of the solution x of $F(x) - u = 0$ can be calculated with bisection, or Newton's method, or the secant method ($\longrightarrow$ Appendix C.1).

Alternatively the inversion $x = F^{-1}(u)$ can be approximated by a suitably constructed function $G(u)$ with

$$G(u) \approx F^{-1}(u)\,.$$

[3] Also called *inversion sampling*.

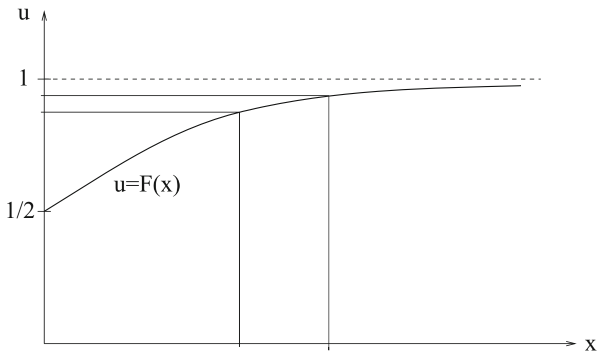

Fig. 2.4 Normal distribution; small changes in u can lead to large changes in x

Then only $x = G(u)$ needs to be evaluated. Constructing such an approximation formula G, it is important to realize that $F^{-1}(u)$ has "vertical" tangents at $u = 1$ (horizontal in Fig. 2.4). The pole behavior must be reproduced correctly by an approximating function G. This suggests to use rational approximation ($\longrightarrow$ Appendix C.1). For the Gaussian distribution one incorporates the point symmetry with respect to $(u, x) = (\frac{1}{2}, 0)$, and the pole at $u = 1$ (and hence at $u = 0$) in the *ansatz* for G ($\longrightarrow$ Exercise 2.10). Rational approximation of $F^{-1}(u)$ with a sufficiently large number of terms leads to high accuracy [278]. The formulas are given in Appendix E.2.

2.2.2 *Transformation in* $\mathbb{R}^1$

Another class of methods uses transformations between random variables. We start the discussion with the scalar case. If we have a random variable X with known density and distribution, what can we say about the density and distribution of a transformed $h(X)$?

Theorem 2.9 (Transformation in Scalar Case) *Suppose X is a random variable with density $f(x)$ and distribution $F(x)$. Further assume $h : S \longrightarrow B$ with $S, B \subseteq \mathbb{R}$, where S is the support[4] of $f(x)$, and let h be strictly monotonic.*

(a) Then $Y := h(X)$ is a random variable. Its distribution F_Y is

$$F_Y(y) = F(h^{-1}(y)) \quad \textit{in case } h' > 0\,,$$
$$F_Y(y) = 1 - F(h^{-1}(y)) \quad \textit{in case } h' < 0\,.$$

[4] f is zero outside S. (In this section, S is no asset price.) Use Theorem 2.9 to check the derivation of f_{GBM} out of $\hat{f}$ in Sect. 1.8.2.

(b) If h^{-1} is absolutely continuous then for almost all y the density of $h(X)$ is

$$f(h^{-1}(y)) \left| \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y} \right| . \tag{2.6}$$

Proof

(a) For $h' > 0$ we have $\mathsf{P}(h(X) \le y) = \mathsf{P}(X \le h^{-1}(y)) = F(h^{-1}(y))$.
(b) For absolutely continuous h^{-1} the density of $Y = h(X)$ is equal to the derivative of the distribution function almost everywhere. Evaluating the derivative $\frac{\mathrm{d}F(h^{-1}(y))}{\mathrm{d}y}$ with the chain rule implies the assertion. The absolute value in (2.6) is necessary such that a positive density comes out in case $h' < 0$. (See for instance [131, Sect. 2.4 C].)

2.2.2.1 Application

Being able to calculate uniform deviates, we start from $X \sim \mathcal{U}[0, 1]$ with the density f of the uniform distribution,

$$f(x) = 1 \;\text{ for }\; 0 \le x \le 1, \;\text{ otherwise } f = 0 .$$

Here the support S is the unit interval. What we need are random numbers Y matching a prespecified target density $g(y)$. It remains to find a transformation h such that the density in (2.6) is identical to $g(y)$,

$$1 \cdot \left| \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y} \right| = g(y) .$$

Then only evaluate $h(X)$.

Example 2.10 (Exponential Distribution) The exponential distribution with parameter $\lambda > 0$ has the density

$$g(y) = \begin{cases} \lambda \mathrm{e}^{-\lambda y} & \text{for } y \ge 0 \\ 0 & \text{for } y < 0 . \end{cases}$$

Here the range B consists of the nonnegative real numbers. The aim is to generate an exponentially distributed random variable Y out of a $\mathcal{U}[0, 1]$-distributed random variable X. To this end define the monotone transformation from the unit interval $S = [0, 1]$ into B by the decreasing function

$$y = h(x) := -\frac{1}{\lambda} \log x$$

with the inverse function $h^{-1}(y) = \mathrm{e}^{-\lambda y}$ for $y \geq 0$. For this h verify

$$f(h^{-1}(y)) \left| \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y} \right| = 1 \cdot \left|(-\lambda)\mathrm{e}^{-\lambda y}\right| = \lambda \mathrm{e}^{-\lambda y} = g(y)$$

as density of $h(X)$. Hence $h(X)$ is distributed exponentially as long as $X \sim \mathcal{U}[0, 1]$.
Application:

In case $U_1, U_2, \ldots$ are nonzero uniform deviates, the numbers $h(U_i)$

$$-\frac{1}{\lambda}\log(U_1), \quad -\frac{1}{\lambda}\log(U_2), \ \ldots$$

are distributed exponentially. This result is similar to that of the inversion. For an application see Exercise 2.11.

2.2.2.2 Attempt to Generate a Normal Distribution

Starting from the uniform distribution ($f = 1$) a transformation $y = h(x)$ is searched such that its density equals that of the standard normal distribution,

$$1 \cdot \left| \frac{\mathrm{d}h^{-1}(y)}{\mathrm{d}y} \right| = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}y^2\right) .$$

This is a differential equation for h^{-1} without analytic solution. As we will see, a transformation can be applied successfully in $\mathbb{R}^2$. To this end we need a generalization of the scalar transformation of Theorem 2.9 into $\mathbb{R}^n$.

2.2.3 *Transformations in* $\mathbb{R}^n$

The generalization of Theorem 2.9 to the vector case is

Theorem 2.11 (Transformation in Vector Case) *Suppose X is a random variable in* $\mathbb{R}^n$ *with density* $f(x) > 0$ *on the support S. The transformation* $h : S \to B,\ S, B \subseteq \mathbb{R}^n$ *is assumed to be invertible and the inverse be continuously differentiable on B.* $Y := h(X)$ *is the transformed random variable. Then Y has the density*

$$f(h^{-1}(y)) \left| \frac{\partial(x_1, \ldots, x_n)}{\partial(y_1, \ldots, y_n)} \right|, \quad y \in B\,, \tag{2.7}$$

where $x = h^{-1}(y)$ *and* $\frac{\partial(x_1,\ldots,x_n)}{\partial(y_1,\ldots,y_n)}$ *is the determinant of the Jacobian matrix of all first-order derivatives of* $h^{-1}(y)$.
(Theorem 4.2 in [103])

2.2.4 Acceptance-Rejection Method

An acceptance-rejection method[5] is based on the following facts: Let f be a density function on the support $S \subset \mathbb{R}$ and $\mathcal{A}_f$ the area between the x-axis and the graph of f. Assume two random variables U and X independent of each other with $U \sim \mathcal{U}[0,1]$ and X distributed with density f. Then the points

$$(x, y) := (X,\ U \cdot f(X))$$

are distributed uniformly on $\mathcal{A}_f$. And vice versa, the x-coordinates of uniformly distributed points on $\mathcal{A}_f$ are f-distributed. This is illustrated in Fig. 2.5 for the normal distribution. If one cuts off a piece of the area $\mathcal{A}_f$, then the remaining points are still distributed uniformly. This is exploited by rejection methods.

The aim is to calculate f-distributed random numbers; the density f is the target distribution. Let g be another density on S, and assume for a constant $c \geq 1$

$$f(x) \leq c\, g(x) \quad \text{for all } x \in S\,.$$

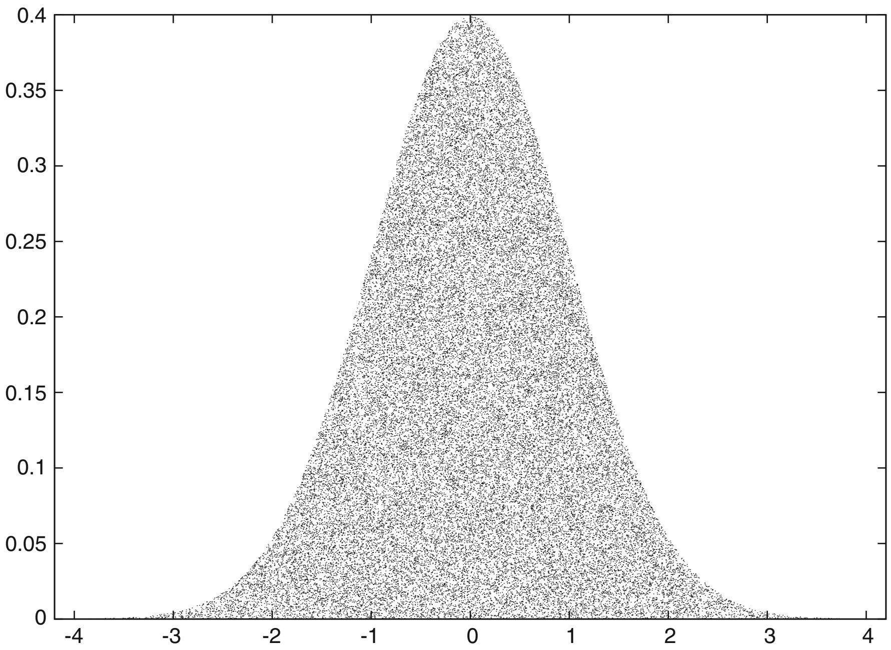

Fig. 2.5 Fifty thousand points $(X,\ Uf(X))$, with $X \sim \mathcal{N}(0,1)$, $U \sim \mathcal{U}[0,1]$. The normal density f of X is visible as envelope

[5]Shortly: rejection method, or rejection sampling.

The function cg is major to f, and the set $\mathcal{A}_f$ is subset of the area $\mathcal{A}_{cg}$ underneath the graph of cg. A rejection algorithm assumes that g-distributed x-samples can be calculated easily. Then the points $(x, ucg(x))$ are distributed uniformly on $\mathcal{A}_{cg}$. Cutting off the part of $\mathcal{A}_{cg}$ above $\mathcal{A}_f$ means to reject points with $ucg(x) > f(x)$. The x-coordinates of the remaining points with $ucg(x) \le f(x)$ are accepted and are distributed as desired.

Algorithm 2.12 (Rejection Method)

```
Repeat:
    $x :=$ random number distributed with density $g$ ,
    $u :=$ random number $\sim \mathcal{U}[0,1]$ independent of $x$ ,
  until   $u\,c\,g(x) \le f(x)$ .
return: $x$
```

As an application of the rejection method consider the Laplace density $g(x) := \frac{1}{2}\exp(-|x|)$ and the standard normal density f, see Exercises 2.9 and 2.12.[6]

2.3 Normally Distributed Random Variables

In this section the focus is on generating normal variates. Fist we describe the fundamental approach of Box and Muller, which applies the transformation method in $\mathbb{R}^2$ to generate Gaussian random numbers.[7]

2.3.1 *Method of Box and Muller*

To apply Theorem 2.11 we start with the unit square $S := [0,1]^2$ and the density (2.7) of the bivariate uniform distribution. The transformation is

$$\begin{aligned} y_1 &= \sqrt{-2\log x_1}\ \cos 2\pi x_2 =: h_1(x_1, x_2) \\ y_2 &= \sqrt{-2\log x_1}\ \sin 2\pi x_2 =: h_2(x_1, x_2)\,, \end{aligned} \tag{2.8}$$

$h(x)$ is defined on $[0,1]^2$ with values in $\mathbb{R}^2$. Its inverse function h^{-1} is given by

$$\begin{aligned} x_1 &= \exp\left\{-\frac{1}{2}(y_1^2 + y_2^2)\right\} \\ x_2 &= \frac{1}{2\pi}\arctan\frac{y_2}{y_1} \end{aligned}$$

[6]Colored in Topic 3 of the *Topics fCF*.

[7]Inversion is one of several valid alternatives. See also the Notes on this section.

where we take the main branch of arctan. The determinant of the Jacobian matrix is

$$\frac{\partial(x_1,x_2)}{\partial(y_1,y_2)} = \det\begin{pmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{pmatrix} =$$

$$= \frac{1}{2\pi}\exp\left\{-\frac{1}{2}(y_1^2+y_2^2)\right\}\left(-y_1\frac{1}{1+\frac{y_2^2}{y_1^2}}\frac{1}{y_1} - y_2\frac{1}{1+\frac{y_2^2}{y_1^2}}\frac{y_2}{y_1^2}\right)$$

$$= -\frac{1}{2\pi}\exp\left\{-\frac{1}{2}(y_1^2+y_2^2)\right\} .$$

This shows that $\left|\frac{\partial(x_1,x_2)}{\partial(y_1,y_2)}\right|$ is the density (2.7) of the bivariate standard normal distribution. Since this density is the product of the two one-dimensional densities,

$$\left|\frac{\partial(x_1,x_2)}{\partial(y_1,y_2)}\right| = \left[\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}y_1^2\right)\right]\cdot\left[\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}y_2^2\right)\right] ,$$

the two components of the vector y are independent. So, when the components of the vector X are $\sim \mathcal{U}[0,1]$, the vector $h(X)$ consists of two independent standard normal variates. Let us summarize the application of this transformation:

Algorithm 2.13 (Box–Muller)

```
Generate $U_1 \sim \mathcal{U}[0,1]$ and $U_2 \sim \mathcal{U}[0,1]$.
$\theta := 2\pi U_2 , \quad \rho := \sqrt{-2\log U_1}$ .
$Z_1 := \rho\cos\theta$ is a normal variate (as well as $Z_2 := \rho\sin\theta$).
```

The variables U_1, U_2 stand for the components of X. Each application of the algorithm provides two standard normal variates. Note that a line structure in $[0,1]^2$ as in Example 2.5 is mapped to curves in the (Z_1, Z_2)-plane. This underlines the importance of excluding an evident line structure.

2.3.2 Variant of Marsaglia

The variant of Marsaglia prepares the input in Algorithm 2.13 such that trigonometric functions are avoided. For $U \sim \mathcal{U}[0,1]$ we have $V := 2U - 1 \sim \mathcal{U}[-1,1]$. (Temporarily we misuse also the financial variable V for local purposes.) Two values V_1, V_2 calculated in this way define a point in the (V_1, V_2)-plane. Only points within the unit disk $\mathcal{D}$ are accepted:

$$\mathcal{D} := \{ (V_1, V_2) \mid V_1^2 + V_2^2 < 1 \} ; \text{ accept only } (V_1, V_2) \in \mathcal{D} .$$

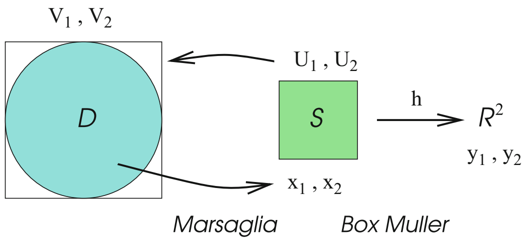

Fig. 2.6 Transformations of the Box–Muller–Marsaglia approach, schematically

In case of rejection both values V_1, V_2 must be rejected. As a result, the surviving (V_1, V_2) are uniformly distributed on $\mathcal{D}$ with density $f(V_1, V_2) = \frac{1}{\pi}$ for $(V_1, V_2) \in \mathcal{D}$. A transformation from the disk $\mathcal{D}$ into the unit square $S := [0, 1]^2$ is defined by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} V_1^2 + V_2^2 \\ \frac{1}{2\pi}\arg((V_1, V_2)) \end{pmatrix} .$$

That is, the Cartesian coordinates V_1, V_2 on $\mathcal{D}$ are mapped to the squared radius and the normalized angle.[8] For illustration, see Fig. 2.6. These "polar coordinates" (x_1, x_2) are uniformly distributed on S ($\longrightarrow$ Exercise 2.13).

Application

For input in (2.8) use $V_1^2 + V_2^2$ as x_1 and $\frac{1}{2\pi}\arctan\frac{V_2}{V_1}$ as x_2. With these variables the relations

$$\cos 2\pi x_2 = \frac{V_1}{\sqrt{V_1^2 + V_2^2}} , \quad \sin 2\pi x_2 = \frac{V_2}{\sqrt{V_1^2 + V_2^2}} ,$$

hold, which means that it is no longer necessary to evaluate trigonometric functions. The resulting algorithm of Marsaglia has modified the Box–Muller method by constructing input values x_1, x_2 in a clever way.

Algorithm 2.14 (Polar Method)

```
Repeat:
    generate $U_1, U_2 \sim \mathcal{U}[0, 1]$ ;
    calculate $V_1 := 2U_1 - 1$ , $V_2 := 2U_2 - 1$
        until $w := V_1^2 + V_2^2 < 1$ .
$Z_1 := V_1\sqrt{-2\log(w)/w}$
$Z_2 := V_2\sqrt{-2\log(w)/w}$
    are both standard normal variates.
```

[8] $\arg((V_1, V_2)) = \arctan(V_2/V_1)$ with the proper branch.

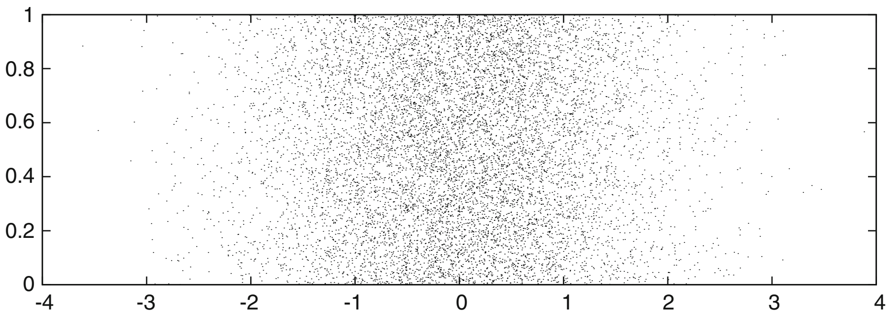

Fig. 2.7 Ten thousand numbers $\sim \mathcal{N}(0, 1)$ (values entered horizontally and separated vertically with distance 10^{-4})

The probability that $w < 1$ holds is given by the ratio of the areas, $\pi/4 = 0.785\ldots$ Hence in about 21% of all $\mathcal{U}[0, 1]$ drawings the (V_1, V_2)-tuple is rejected because of $w \geq 1$. Nevertheless the savings of the trigonometric evaluations makes Marsaglia's polar method more efficient than the Box–Muller method. Figure 2.7 illustrates normally distributed random numbers ($\longrightarrow$ Exercise 2.14).

2.3.3 Ziggurat

A most efficient algorithm for the generation of normal deviates is the *ziggurat* algorithm, which is a rejection method. The setup consists of a kind of horizontal histogram, which covers the area underneath the graph of a monotonically decreasing f. Figure 2.8, which will explained below, may give an impression of the setup.[9] Here f is the standard normal density $f(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}x^2)$. Because of the symmetry of f it suffices to take $x \geq 0$; a random sign (each with probability $\frac{1}{2}$) must be attached in the end.

The histogram-like area consists of N horizontal and parallel segments each *of equal area* A. We label them by i, with $i = 0$ for the bottom layer and $i = N - 1$ for the top layer. The top $N - 1$ segments are rectangles, whereas the lowest segment ($i = 0$) is limited by the infinite tail of f. The lengths of the segments are defined by f, as illustrated in Figs. 2.8 and 2.9. The upper edges of the segments define a major function z with $z(x) \geq f(x)$ for $x \geq 0$. The major z corresponds to cg in Sect. 2.2.4.

The curve of $f(x)$, decreasing for $x > 0$, enters and leaves the layers, which defines the length x_i of the rectangle, as shown in Fig. 2.9. For a chosen value of N, the requirement of equal area A of all segments leads to a system of equations

[9]The shape explains the use of the name ziggurat, which was a terraced pyramid in the ancient world.

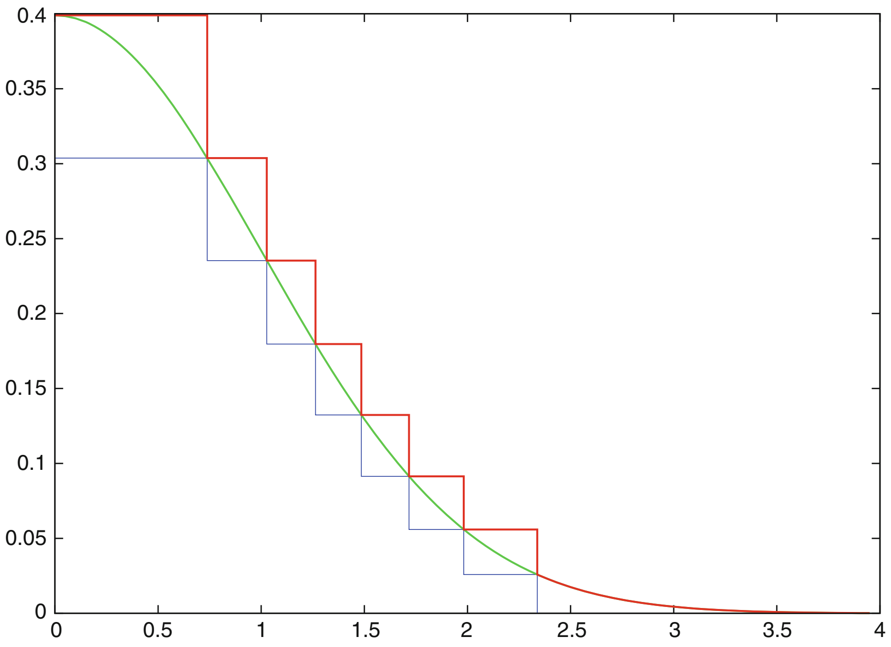

Fig. 2.8 Ziggurat with $N = 8$ layers, for $0 \leq x \leq 4$. The heavy-line zigzag (in *red*) on the *right* and above the graph of the normal density f (in *green*) is the major z with $z(x) \geq f(x)$, which represents the right-hand bound of the horizontal ziggurat boxes. For $N = 8$ the area of each ziggurat segment is 0.070283. The zigzag in *blue* that is below f bounds the area, in which the creation of a normally distributed sample essentially only costs one generation of $U \sim \mathcal{U}[0, 1]$

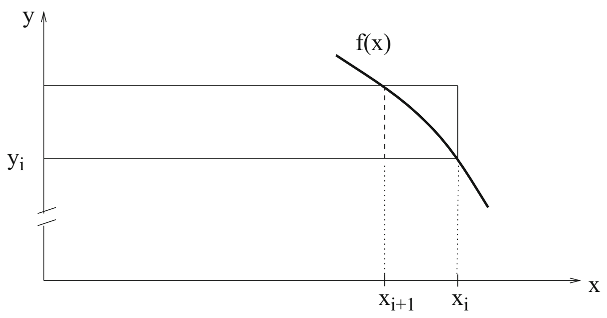

Fig. 2.9 Configuration of the ith layer of the ziggurat, $0 < i < N - 1$, for $x \geq 0$

that defines A and the coordinates (x_i, y_i) of the vertices of the rectangles, where $y_i := f(x_i)$. The coordinates (x_i, y_i) and the value of A are precomputed and stored in a look-up table ($\longrightarrow$ Exercise 2.15). Figure 2.9 illustrates the ith layer ($0 < i < N - 1$). The resulting box consists of two sub-boxes, divided by the coordinate x_{i+1}.

The rejection method needs points (ξ, η) uniformly distributed over the area $\mathcal{A}_z$ underneath the graph of z and above the positive x-axis. In principle, these points are tested for their location relative to f. With the above setup, the check for acceptance or rejection is extremely efficient, because mostly η is not needed explicitly. Since each of the N segments has the same area A, it suffices to draw one of them. Draw the layer i randomly with equal probability $1/N$. Let us first discuss the cases $i > 0$. In rectangle i the next task would be to sample a point (ξ, η), which must be distributed uniformly. Its x-component is given by $\xi := U_1 x_i$, where $U_1 \sim \mathcal{U}[0, 1]$. In case $\xi \leq x_{i+1}$ the point falls in the left-hand part of the rectangle underneath the graph of f, and is accepted. In this case no y-component η is needed! (This does not happen for $i = N - 1$, where $x_N = 0$.) Only in the other case, for $\xi > x_i$, an η is required and f must be evaluated to further test for $\eta \leq f(\xi)$. This is provided by generating a $U_2 \sim \mathcal{U}[0, 1]$ and $\eta := y_i + U_2(y_{i+1} - y_i)$. Acceptance for $\eta \leq f(\xi)$.

The efficiency of the method originates from the fact that the y-component η will be required only in a small portion of samples. In Fig. 2.8 we have chosen $N = 8$ for ease of demonstration. But even for this small value of N the subarea in which no η and no $f(\xi)$ are needed, covers 72.8% of the area underneath z. And when the number N of layers is large, say $N = 256$, the rectangles are narrow, and for $0 < i < N - 1$ the right-hand portions of the rectangles will be much smaller than the left-hand portions. The latter cover the bulk of the area underneath f or z, and there the test for acceptance costs almost nothing: The generated value of U_1 can be compared directly to precomputed ratios x_{i+1}/x_i. In case of acceptance, the output is ξ, and—with attached random sign—the desired number is distributed $\sim \mathcal{N}(0, 1)$. In case of rejection the next i is drawn.

Only the situation of the bottom layer $i = 0$ is more complex. This bottom segment is divided into a rectangle with area $x_1 y_1$, and the infinite tail with $x > x_1$ and area $A - x_1 y_1$. For $i = 0$, the probability of a uniformly sampled point to fall into the rectangle is $x_1 y_1 / A$. So the above simple test can be modified to comparing $\xi := U_1 A / y_1$ to x_1. Accept in case $\xi \leq x_1$. Only in the case $\xi > x_1$ the ziggurat algorithm requires a fallback routine, which resorts to more conventional methods. But this fallback routine for $i = 0$ effects only a tiny part of the overall costs. Even for the small value $N = 8$ of Fig. 2.8, the fallback routine is required only in 2% of all samples. For the tricky implementation of the ziggurat algorithm see [261].

2.3.4 Correlated Random Variables

The above algorithms provide independent normal deviates. In many applications random variables are required that depend on each other in a prescribed way. Let us first recall the general n-dimensional density function.

Multivariate normal distribution (notations):

$$X = (X_1, \ldots, X_n), \quad \mu = \mathsf{E}X = (\mathsf{E}X_1, \ldots, \mathsf{E}X_n)$$

The covariance matrix (B.8) of X is denoted Σ, and has elements

$$\Sigma_{ij} = (\mathsf{Cov}X)_{ij} := \mathsf{E}\left((X_i - \mu_i)(X_j - \mu_j)\right) , \quad \sigma_i^2 = \Sigma_{ii} ,$$

for $i, j = 1, \ldots, n$. Using this notation, the correlation coefficients are

$$\rho_{ij} := \frac{\Sigma_{ij}}{\sigma_i \sigma_j} \qquad (\Rightarrow \ \rho_{ii} = 1) , \tag{2.9}$$

which set up the correlation matrix. The correlation matrix is a scaled version of Σ. The density function $f(x_1, \ldots, x_n)$ corresponding to $\mathcal{N}(\mu, \Sigma)$ is

$$f(x) = \frac{1}{(2\pi)^{n/2}} \frac{1}{(\det \Sigma)^{1/2}} \exp\left\{ -\frac{1}{2}(x - \mu)^{tr} \Sigma^{-1} (x - \mu) \right\} . \tag{2.10}$$

By theory, a covariance matrix (or correlation matrix) Σ is symmetric, and positive semidefinite. If in practice a matrix $\tilde{\Sigma}$ is corrupted by insufficient data, a close matrix Σ can be calculated with the features of a covariance matrix [184, 200]. In case $\det \Sigma \neq 0$ the matrix Σ is positive definite, which we assume now.

Below we shall need a factorization of Σ into $\Sigma = AA^{tr}$. From numerical mathematics we know that for symmetric positive definite matrices Σ the Cholesky decomposition $\Sigma = LL^{tr}$ exists, with a lower triangular matrix L ($\longrightarrow$ Appendix C.1). There are numerous factorizations $\Sigma = AA^{tr}$ other than Cholesky. A more involved factorization of Σ is the principal component analysis, which is based on eigenvectors ($\longrightarrow$ Exercise 2.16).

2.3.4.1 Transformation

Suppose $Z \sim \mathcal{N}(0, I)$ and $x = Az$, $A \in \mathbb{R}^{n \times n}$, where z is a realization of Z, 0 is the zero vector, and I the identity matrix. We apply Theorem 2.11 with $X = h(Z) := AZ$. Accordingly, the density of X is

$$\begin{aligned} f(A^{-1}x)\,|\det(A^{-1})| &= \frac{1}{(2\pi)^{n/2}} \exp\left\{ -\frac{1}{2}(A^{-1}x)^{tr}(A^{-1}x) \right\} \frac{1}{|\det(A)|} \\ &= \frac{1}{(2\pi)^{n/2}} \frac{1}{|\det(A)|} \exp\left\{ -\frac{1}{2} x^{tr} (AA^{tr})^{-1} x \right\} \end{aligned}$$

for arbitrary nonsingular matrices A. To complete the transformation,[10] we need a matrix A such that $\Sigma = AA^{tr}$. Then $|\det A| = (\det \Sigma)^{1/2}$, and the densities with the respect to x and z are converted correctly. In view of the general density $f(x)$ recalled in (2.10), AZ is normally distributed with $AZ \sim \mathcal{N}(0, AA^{tr})$, and hence the factorization $\Sigma = AA^{tr}$ implies

$$AZ \sim \mathcal{N}(0, \Sigma)\,.$$

Finally, translation with vector μ implies

$$\mu + AZ \sim \mathcal{N}(\mu, \Sigma)\,. \tag{2.11}$$

2.3.4.2 Application

Suppose we need a normal variate $X \sim \mathcal{N}(\mu, \Sigma)$ for given mean vector μ and covariance matrix Σ. This is most conveniently based on the Cholesky decomposition of Σ. Accordingly, the desired random variable can be calculated with the following algorithm:

Algorithm 2.15 (Correlated Normal Random Variables)

```
Calculate $A$ via the Cholesky decomposition $AA^{tr} = \Sigma$.
Calculate $Z \sim \mathcal{N}(0, I)$ componentwise
    by $Z_i \sim \mathcal{N}(0, 1)$ for $i = 1, \ldots, n$,
    for instance, with Marsaglia's polar algorithm.
$\mu + AZ$ has the desired distribution $\sim \mathcal{N}(\mu, \Sigma)$.
```

Special case $n = 2$: In this case, in view of (2.9), only one correlation number is involved, namely, $\rho := \rho_{12} = \rho_{21}$, and the covariance matrix must be of the form

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}\,. \tag{2.12}$$

In this two-dimensional situation it makes sense to carry out the Cholesky decomposition analytically ($\longrightarrow$ Exercise 2.17). Figure 2.10 illustrates a highly correlated two-dimensional situation, with $\rho = 0.85$. An example based on (2.12) is (3.35).

[10]Check this by applying Theorem 2.11.

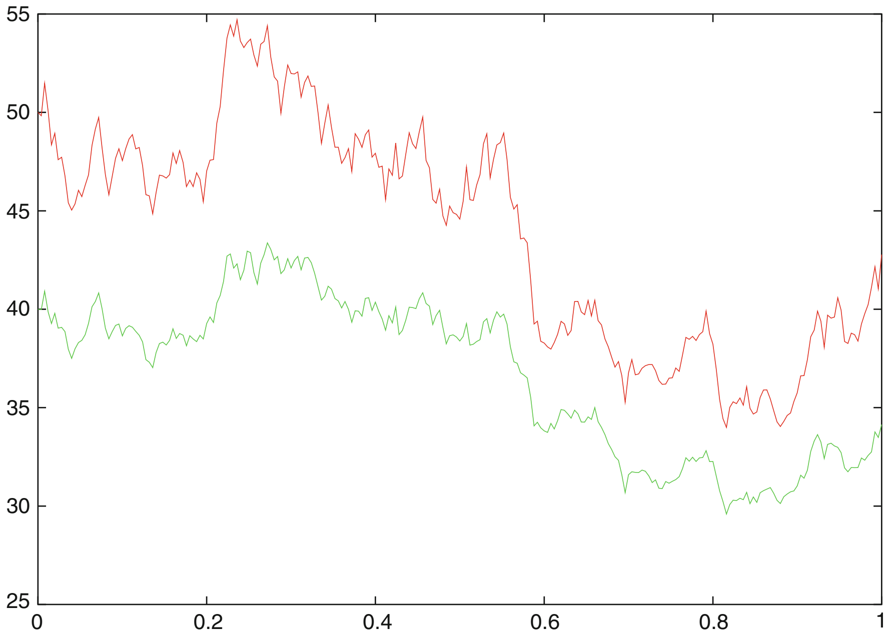

Fig. 2.10 Simulation of a correlated vector process with two components, and $\mu = 0.05$, $\sigma_1 = 0.3$, $\sigma_2 = 0.2$, $\rho = 0.85$, $\Delta t = 1/250$

2.4 Monte Carlo Integration

A classic application of random numbers is Monte Carlo integration. The discussion in this section will serve as background for Quasi Monte Carlo, a topic of the following Sect. 2.5.

Let us begin with the one-dimensional situation. Assume a probability distribution with density f. Then the expectation of a function g is

$$\mathsf{E}(g) = \int_{-\infty}^{\infty} g(x)f(x)\,\mathrm{d}x\,,$$

compare (B.4). For a definite integral on an interval $\mathcal{D} = [a, b]$, we use the uniform distribution with density

$$f = \frac{1}{b-a} \cdot \mathbf{1}_{\mathcal{D}} = \frac{1}{\lambda_1(\mathcal{D})} \cdot \mathbf{1}_{\mathcal{D}}\,,$$

where $\lambda_1(\mathcal{D})$ denotes the length of the interval $\mathcal{D}$ and $\mathbf{1}_{\mathcal{D}}$ the identity on $\mathcal{D}$. This leads to

$$\mathsf{E}(g) = \frac{1}{\lambda_1(\mathcal{D})} \int\limits_a^b g(x)\,\mathrm{d}x\,,$$

or

$$\int\limits_a^b g(x)\,\mathrm{d}x = \lambda_1(\mathcal{D}) \cdot \mathsf{E}(g)\,,$$

the basis of *Monte Carlo integration*. It remains to approximate $\mathsf{E}(g)$. For independent samples $x_k \sim \mathcal{U}[a,b]$, $k = 1,2,\ldots$, apply the law of large numbers ($\longrightarrow$ Appendix B.1) to establish the estimator

$$\frac{1}{N}\sum_{k=1}^{N} g(x_k)$$

as approximation to $\mathsf{E}(g)$. The approximation improves as the number of trials N goes to infinity; the error is characterized by the central limit theorem.

This principle of Monte Carlo integration extends to the higher-dimensional case. Let $\mathcal{D} \subset \mathbb{R}^m$ be a domain on which the integral

$$\int_{\mathcal{D}} g(x)\,\mathrm{d}x$$

is to be calculated. For example, on the hypercube $\mathcal{D} = [0,1]^m$. Such integrals occur in finance, for example, when mortgage-backed securities (CMO, collateralized mortgage obligations) are valuated [64]. The classic or *stochastic Monte Carlo integration* draws random samples $x_1,\ldots,x_N \in \mathcal{D}$ which should be independent and uniformly distributed. Then

$$\theta_N := \lambda_m(\mathcal{D})\frac{1}{N}\sum_{k=1}^{N} g(x_k) \tag{2.13}$$

is an approximation of the integral. Here $\lambda_m := \lambda_m(\mathcal{D})$ is the volume of $\mathcal{D}$ (or the m-dimensional Lebesgue measure [286]). We assume λ_m to be finite. From the law of large numbers follows convergence of θ_N to $\lambda_m \mathsf{E}(g) = \int_{\mathcal{D}} g(x)\,\mathrm{d}x$ for $N \to \infty$. The variance of the error

$$\delta_N := \int_{\mathcal{D}} g(x)\,\mathrm{d}x - \theta_N$$

Table 2.1 Comparison of different convergence rates to zero

N	$\frac{1}{\sqrt{N}}$	$\sqrt{\frac{\log\log N}{N}}$	$\frac{\log N}{N}$	$\frac{(\log N)^2}{N}$	$\frac{(\log N)^3}{N}$
10^1	0.31622777	0.28879620	0.23025851	0.53018981	1.22080716
10^2	0.10000000	0.12357911	0.04605170	0.21207592	0.97664572
10^3	0.03162278	0.04396186	0.00690776	0.04771708	0.32961793
10^4	0.01000000	0.01490076	0.00092103	0.00848304	0.07813166
10^5	0.00316228	0.00494315	0.00011513	0.00132547	0.01526009
10^6	0.00100000	0.00162043	0.00001382	0.00019087	0.00263694
10^7	0.00031623	0.00052725	0.00000161	0.00002598	0.00041874
10^8	0.00010000	0.00017069	0.00000018	0.00000339	0.00006251
10^9	0.00003162	0.00005506	0.00000002	0.00000043	0.00000890

satisfies

$$\mathsf{Var}(\delta_N) = \mathsf{E}(\delta_N^2) - (\mathsf{E}(\delta_N))^2 = \frac{\sigma^2(g)}{N}(\lambda_m)^2\,, \tag{2.14}$$

with the variance of g

$$\sigma^2(g) := \frac{1}{\lambda_m}\int_{\mathcal{D}} g(x)^2\,\mathrm{d}x - \frac{1}{\lambda_m^2}\left(\int_{\mathcal{D}} g(x)\,\mathrm{d}x\right)^2\,. \tag{2.15}$$

Hence the standard deviation of the error δ_N tends to 0 with the order $O(N^{-1/2})$. This result follows from the central limit theorem or from other arguments ($\longrightarrow$ Exercise 2.18). The deficiency of the order $O(N^{-1/2})$ is the slow convergence ($\longrightarrow$ Exercise 2.19 and the second column in Table 2.1). To reach an absolute error of the order ε, Eq. (2.14) tells that the sample size is $N = O(\varepsilon^{-2})$. To improve the accuracy by a factor of 10, the costs (that is the number of trials, N) increase by a factor of 100. Another disadvantage is the lack of a genuine error *bound*. The probabilistic error of (2.14) does not rule out the risk that the result may be completely wrong. The $\sigma^2(g)$ in (2.15) is not known and must be approximated. Monte Carlo integration responds sensitively to changes of the initial state of the used random-number generator. This may be explained by the potential clustering of random points.

In many applications the above deficiencies are balanced by two good features of Monte Carlo integration: A first advantage is that the order $O(N^{-1/2})$ of the error holds independently of the dimension m. Another good feature is that the integrands g need not be smooth, square integrability suffices ($g \in \mathcal{L}^2$, see Appendix C.3).

So far we have described the basic version of Monte Carlo integration, stressing the slow decline of the probabilistic error with growing N. The variance of the error δ can also be diminished by decreasing the numerator in (2.14). This variance of the problem can be reduced by suitable methods. (We will come back to this issue in Sect. 3.5.4.)

We conclude the excursion into the stochastic Monte Carlo integration with the variant for those cases in which $\lambda_m(\mathcal{D})$ is hard to calculate. For $\mathcal{D} \subseteq [0,1]^m$ and $x_1, \ldots, x_N \sim \mathcal{U}[0,1]^m$ use

$$\int_{\mathcal{D}} f(x)\,\mathrm{d}x \approx \frac{1}{N} \sum_{\substack{k=1 \\ x_k \in \mathcal{D}}}^{N} f(x_k)\,. \tag{2.16}$$

For the integral (1.66) with density f_{GBM}, see Sect. 3.5.

2.5 Sequences of Numbers with Low Discrepancy

One difficulty with random numbers is that they may fail to distribute uniformly. Here, "uniform" is not meant in the stochastic sense of a distribution $\sim \mathcal{U}[0,1]$, but has the meaning of an equidistributedness that avoids extreme clustering or holes. The aim is to generate numbers for which the deviation from uniformity is minimal. This deviation is called "discrepancy." Another objective is to obtain good convergence for some important applications.

2.5.1 Discrepancy

The bad convergence behavior of the stochastic Monte Carlo integration is not inevitable. For example, for $m = 1$ and $\mathcal{D} = [0,1]$ an equidistant x-grid with mesh size $1/N$ leads to a formula (2.13) that resembles the trapezoidal sum [(C.2) in Appendix C.1]. For smooth g, the order of the error is at least $O(N^{-1})$. (Why?) But such a grid-based evaluation procedure is somewhat inflexible because the grid must be prescribed in advance and the number N that matches the desired accuracy is unknown beforehand. In contrast, the free placing of sample points with Monte Carlo integration can be performed until some termination criterion is met. It would be desirable to find a compromise in placing sample points such that the fineness advances but clustering is avoided. The sample points should fill the integration domain $\mathcal{D}$ as uniformly as possible. To this end we require a measure of the *equidistributedness*.[11]

For $m \geq 1$ let $Q \subseteq [0,1]^m$ be an arbitrary axially parallel m-dimensional box (hyperrectangle) in the unit cube $[0,1]^m$ of $\mathbb{R}^m$. That is, Q is a product of m intervals. Suppose a set of points $x_1, \ldots, x_N \in [0,1]^m$. The decisive idea behind discrepancy is that for an evenly distributed point set, the fraction of the points lying within the box Q should correspond to the volume of the box (see Fig. 2.11). Let # denote the

[11]The deterministic term "equidistributed" is not to be confused with the probabilistic "uniformly distributed".

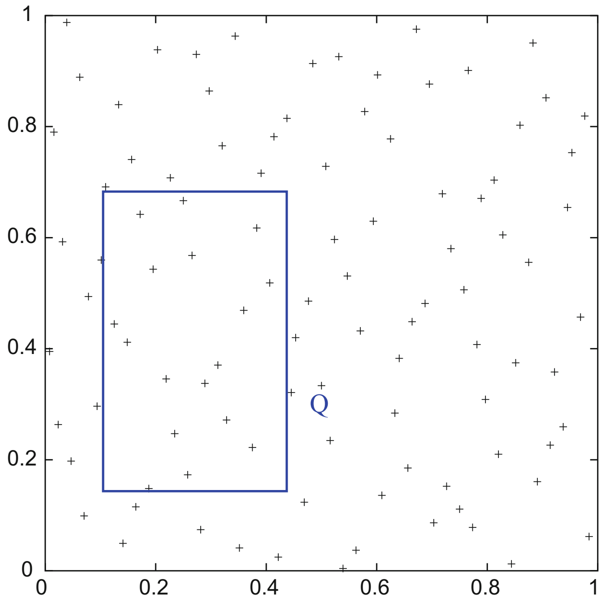

Fig. 2.11 On the idea of discrepancy, here for $m = 2$

number of points, then the goal is

$$\frac{\#\text{ of } x_i \in Q}{\#\text{ of all points in } [0,1]^m} \approx \frac{\text{vol}(Q)}{\text{vol}([0,1]^m)}$$

for as many boxes Q as possible. This leads to the following definition:

Definition 2.16 (Discrepancy) The discrepancy of the point set $\{x_1, \ldots, x_N\} \subset [0,1]^m$ is

$$D_N := \sup_Q \left| \frac{\#\ \text{of } x_i \in Q}{N} - \text{vol}(Q) \right| .$$

Obviously, Figs. 2.1 and 2.2 allow to construct relatively large rectangles Q such that no points land on Q. Then D_N will not become small for increasing N. The more evenly the points of a sequence are distributed, the closer the discrepancy D_N is to zero. The criterion

$$\lim_{N\to\infty} D_N = 0$$

will characterize equidistributed points. Here D_N refers to the first N points of a sequence of points (x_i), $i \geq 1$.

Analogously the variant D_N^* *(star discrepancy)* is obtained when the set of boxes is restricted to those Q^*, for which one corner is the origin:

$$Q^* = \prod_{i=1}^{m} [0, y_i)$$

where $y \in \mathbb{R}^m$ denotes the corner diagonally opposite the origin. The discrepancies D_N and D_N^* satisfy [$\longrightarrow$ Exercise 2.20(b)]

$$D_N^* \le D_N \le 2^m D_N^* .$$

The discrepancy allows to find a deterministic bound on the error δ_N of Monte Carlo integration,

$$|\delta_N| \le \mathcal{V}(g) D_N^* ; \tag{2.17}$$

here $\mathcal{V}(g)$ is the variation[12] of the function g with $\mathcal{V}(g) < \infty$, and the domain of integration is $\mathcal{D} = [0, 1]^m$ [280, 286, 363]. This result is known as Theorem of Koksma and Hlawka. The bound in (2.17) underlines the importance to find numbers $x_1, \dots, x_N$ with small value of the discrepancy D_N. After all, a set of N *randomly* chosen points satisfies

$$\mathsf{E}(D_N) = O\left(\sqrt{\frac{\log \log N}{N}}\right) .$$

This is in accordance with the probabilistic $O(N^{-1/2})$ law. The order of magnitude of these numbers is shown in Table 2.1 (third column).

Definition 2.17 (Low-Discrepancy Point Sequence) A sequence of points or numbers $x_1, x_2, \dots, x_N, \dots \in [0, 1]^m$ is called low-discrepancy sequence if

$$D_N \le C_m \frac{(\log N)^m}{N} \tag{2.18}$$

for a constant C_m independent of N.

Deterministic sequences of numbers satisfying (2.18) are also called *quasi*-random numbers, although they are fully deterministic. Table 2.1 reports on the orders of magnitude. Since $\log(N)$ grows only modestly, a low discrepancy essentially means $D_N \approx O(N^{-1})$ as long as the dimension m is small. The Eq. (2.18)

[12] As in Sect. 1.6.2.

expresses some dependence on the dimension m, contrary to Monte Carlo methods. But the dependence on m in (2.18) is less stringent than with classic MC quadrature.

2.5.2 Examples of Low-Discrepancy Sequences

In the one-dimensional case ($m = 1$) the point set

$$x_i = \frac{2i-1}{2N}, \quad i = 1, \ldots, N \tag{2.19}$$

has the value $D_N^* = \frac{1}{2N}$; this value can not be improved [$\longrightarrow$ Exercise 2.20(c)]. The monotone sequence (2.19) can be applied only when a reasonable N is known and fixed; for $N \to \infty$ the x_i would be newly placed and an integrand g evaluated again. Since N is large, it is essential that the previously calculated results can be used when N is growing. This means that the points $x_1, x_2, \ldots$ must be placed "dynamically" so that they are preserved and the fineness improves when N grows. This is achieved by the sequence

$$\frac{1}{2}, \ \frac{1}{4}, \frac{3}{4}, \ \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \ \frac{1}{16}, \ldots$$

This sequence is known as van der Corput sequence. To motivate such a dynamical placing of points imagine that you are searching for some item in the interval $[0, 1]$ (or in the cube $[0, 1]^m$). The searching must be fast and successful, and is terminated as soon as the object is found. This defines N dynamically by the process.

The formula that defines the van der Corput sequence can be formulated as algorithm. Let us study an example, say, $x_6 = \frac{3}{8}$. The index $i = 6$ is written as binary number

$$6 = (110)_2 =: (d_2\, d_1\, d_0)_2 \quad \text{with} \quad d_i \in \{0, 1\}\,.$$

Then reverse the binary digits and put the radix point in front of the sequence:

$$(.\, d_0\, d_1\, d_2)_2 = \frac{d_0}{2} + \frac{d_1}{2^2} + \frac{d_3}{2^3} = \frac{1}{2^2} + \frac{1}{2^3} = \frac{3}{8}$$

If this is done for all indices $i = 1, 2, 3, \ldots$ the van der Corput sequence $x_1, x_2, x_3, \ldots$ results. These numbers can be defined with the following function:

Definition 2.18 (Radical-Inverse Function) For $i = 1, 2, \ldots$ let j be given by the expansion in base b (integer ≥ 2)

$$i = \sum_{k=0}^{j} d_k b^k\,,$$

with digits $d_k \in \{0, 1, \ldots, b-1\}$, which depend on b, i. Then the radical-inverse function is defined by

$$\phi_b(i) := \sum_{k=0}^{j} d_k b^{-k-1} .$$

The function $\phi_b(i)$ is the digit-reversed fraction of i. This mapping can be seen as reflecting with respect to the radix point. To each index i a rational number $\phi_b(i)$ in the interval $0 < x < 1$ is assigned. Every time the number of digits j increases by one, the mesh becomes finer by a factor $1/b$. This means that the algorithm fills all mesh points on the sequence of meshes with increasing fineness ($\longrightarrow$ Exercise 2.21). Van der Corput's sequence is obtained by

$$x_i := \phi_2(i) .$$

The radical-inverse function can be applied to construct points x_i in the m-dimensional cube $[0, 1]^m$. A simple construction is the Halton sequence.

Definition 2.19 (Halton Sequence) Let $p_1, \ldots, p_m$ be pairwise prime integers. The Halton sequence is defined as the sequence of vectors

$$x_i := \left(\phi_{p_1}(i), \ldots, \phi_{p_m}(i)\right) , \quad i = 1, 2, \ldots$$

Usually one takes $p_1, \ldots, p_m$ as the first m prime numbers. Figure 2.12 shows for $m = 2$ and $p_1 = 2$, $p_2 = 3$ the first 10,000 Halton points. Compared to the pseudo-random points of Fig. 2.3, the Halton points are distributed more evenly.

Halton sequences x_i of Definition 2.19 are easily constructed, but fail to be equidistributed when the dimension m is high, see [155], Sect. 5.2. Then correlations between the radical-inverse functions for different dimensions are observed. This problem can be cured with a simple modification of the Halton sequence, namely, by using only every lth Halton number [227]. The leap l is a prime different from all bases $p_1, \ldots, p_m$. The result is the "Halton sequence leaped"

$$x_k := \left(\phi_{p_1}(lk), \ldots, \phi_{p_m}(lk)\right) , \quad k = 1, 2, \ldots \tag{2.20}$$

This modification has shown good performance for dimensions at least up to $m = 400$. As reported in [227], $l = 409$ is one example of a good leap value.

Other sequences with low discrepancy have been constructed. These include the sequences developed by Sobol, Faure and Niederreiter, see [280, 286, 306]. All these sequences satisfy

$$D_N^* \le C_m \frac{(\log N)^m}{N} + O\left(\frac{(\log N)^{m-1}}{N}\right) .$$

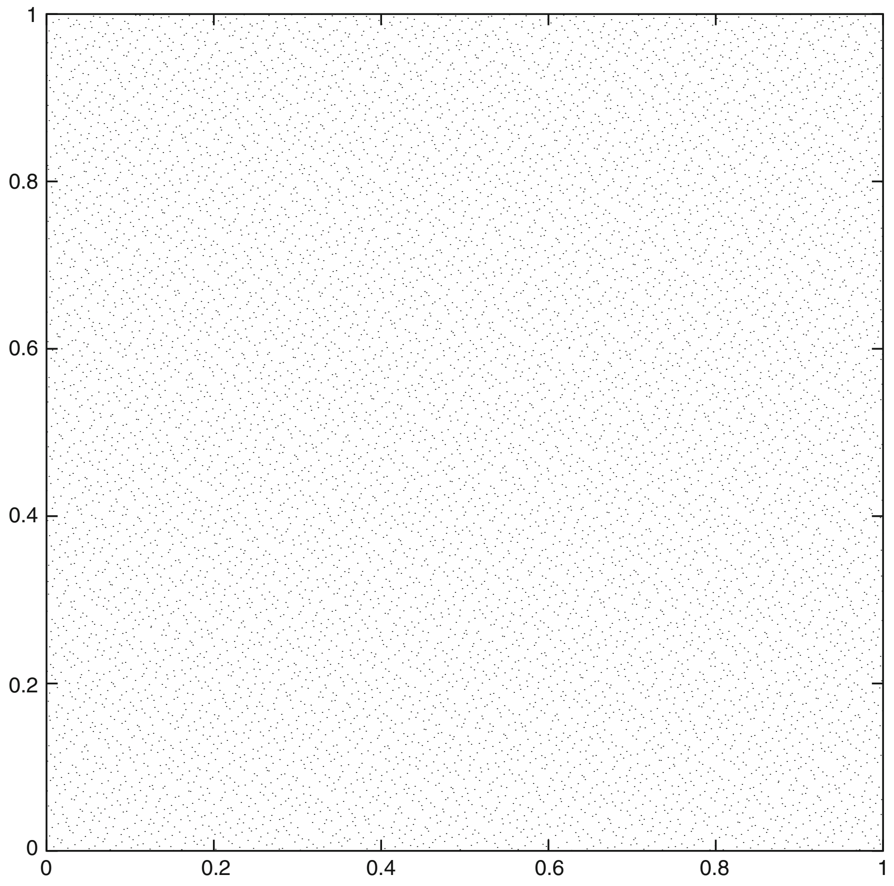

Fig. 2.12 Ten thousand Halton points from Definition 2.19, with $p_1 = 2, p_2 = 3$

Table 2.1 shows how fast the relevant terms $(\log N)^m/N$ tend to zero. If m is large, extremely large values of the denominator N are needed before the terms become small. But it is assumed that the bounds are unrealistically large and overestimate the real error.

Quasi Monte Carlo (QMC) methods approximate the integrals with the arithmetic mean θ_N of (2.13), but use low-discrepancy numbers x_i instead of random numbers. QMC is a *deterministic* method. Practical experience with low-discrepancy sequences are better than might be expected from the bounds known so far. This also holds for the bound (2.17) by Koksma and Hlawka; apparently a large class of functions g satisfy $|\delta_N| \ll \mathcal{V}(g)D_N^*$, see [343].

2.6 Notes and Comments

On Sect. 2.1

The linear congruential method is sometimes called Lehmer generator. Easily accessible and popular generators are RAN1 and RAN2 from [306]. Further references on linear congruential generators include [239, 259, 286, 317]. Example 2.4 is from [130], and Example 2.5 from [317]. Nonlinear congruential generators are of the form

$$N_i = f(N_{i-1}) \mod M\,.$$

Hints on the algorithmic implementation are found in [147]. Generally it is advisable to run the generator in integer arithmetic in order to avoid rounding errors that may spoil the period, see [241]. There are multiplicative Fibonacci generators of the form

$$N_{i+1} := N_{i-\nu}N_{i-\mu} \mod M\,.$$

For Fibonacci generators we refer to [54]. The version of (2.5) is a subtractive generator. Additive versions (with a plus sign instead of the minus sign) are used as well [147, 226]. The codes in [306] are recommendable. For simple statistical tests with illustrations see [181].

Hints on parallelization are given in [262]. For example, parallel Fibonacci generators are obtained by different initializing sequences. Marsaglia's KISS-generator *(keep it simple stupid)* combines different methods and reaches long periods. Programs of professional random number generators (RNG) can be found in the internet. Note that computer systems and software packages often provide built-in random number generators. But often these generators are not clearly specified, and should be handled with care.

On Sects. 2.2 and 2.3

The inversion result of Theorem 2.8 can be formulated placing less or no restrictions on F, see [317, p. 59], [103, p. 28], or [238, p. 270]. The generalized inverse of an arbitrary distribution function F is the *quantile function*

$$Q(u) := \inf_{x\in\mathbb{R}}\{\, x \mid F(x) \geq u \,\}\,,$$

also denoted $F^{-1}(u)$.

For the rejection method, $\frac{1}{c}$ is the proportion of samples distributed from g that are accepted. Hence c should be as small as possible with $c \geq \max_x \frac{f(x)}{g(x)}$. Several algorithms are based on the rejection method [103, 130]; for a detailed overview with many references see [103].

The Box–Muller approach was suggested in [45]. Marsaglia's modification was published in a report quoted in [260]. Fast algorithms aside from the ziggurat include the Wallace algorithm [372], which works with a pool of random numbers and suitable transformations. Platform-dependent implementation details place emphasis on the one or the other advantage. A survey on Gaussian random number generators is [355]. For simulating Lévy processes, see [84]. For singular symmetric positive *semi*definite matrices Σ ($x^{tr}\Sigma x \geq 0$ for all x), the Cholesky decomposition can be cured, see [157], or [155].

On Sect. 2.4

The bounds on errors of the Monte Carlo integration refer to arbitrary functions g; for smooth functions better bounds can be expected. In the one-dimensional case the variation is defined as the supremum of $\sum_j |g(t_j) - g(t_{j-1})|$ over all partitions, see Sect. 1.6.2. This definition can be generalized to higher-dimensional cases. A thorough discussion is [285, 286].

An advanced application of Monte Carlo integration uses one or more methods of *reduction of variance*, which allows to improve the accuracy in many cases [130, 167, 234, 238, 286, 306, 324]. For example, the integration domain can be split into subsets *(stratified sampling)* [316]. Another technique is used when for a *control variate* v with $v \approx g$ the exact integral is known. Then g is replaced by $(g - v) + v$ and Monte Carlo integration is applied to $g - v$. Another alternative, the method of *antithetic variates*, will be described in Sect. 3.5.4 together with the control-variate technique.

On Sect. 2.5

Besides the supremum discrepancy of Definition 2.16 the $\mathcal{L}^2$-analogy of an integral version is used. Hints on speed and preliminary comparison are found in [280]. For application on high-dimensional integrals see [296]. For large values of the dimension m, the bound (2.18) takes large values, which might suggest to discard its use. But the notion of an *effective dimension* and practical results give a favorable picture at least for CMO applications of order $m = 360$ [64]. The error bound of Koksma and Hlawka (2.17) is not necessarily recommendable for practical use, see the discussion in [343]. The analogy of the equidistant lattice in (2.19) in higher-dimensional space has unfavorable values of the discrepancy, $D_N = O\left(\frac{1}{\sqrt[m]{N}}\right)$. For $m > 2$ this is worse than Monte Carlo, compare [317]. Monte Carlo does not take advantage of smoothness of integrands. In the case of smooth integrands, sparse-grid approaches are highly competitive. These refined quadrature methods meliorate the *curse of the dimension*, see [148, 149, 312].

Van der Corput sequences can be based also on other bases. Halton's paper is [166]. Computer programs that generate low-discrepancy numbers are available. For example, Sobol numbers are calculated in [306] and Sobol- and Faure numbers in the computer program FINDER [296] and in [354]. At the current state of the art it is open which point set has the smallest discrepancy in the m-dimensional cube. There are generalized Niederreiter sequences, which include Sobol- and Faure sequences as special cases [354]. In several applications deterministic Monte Carlo seems to be superior to stochastic Monte Carlo [295]. A comparison based on finance applications has shown good performance of Sobol numbers; in [206] Sobol numbers are outperformed by Halton sequences leaped (2.20). Niederreiter and Jau-Shyong Shiue [287] and Chap. 5 in [155] provide more discussion and many references.

Besides volume integration, Monte Carlo is needed to integrate over possibly high-dimensional probability distributions. Drawing samples from the required distribution can be done by running a cleverly constructed Markov chain. This kind of method is called Markov Chain Monte Carlo (MCMC). That is, a chain of random variables $X_0, X_1, X_2, \ldots$ is constructed where for given X_j the next state X_{j+1} does not depend on the history of the chain $X_0, X_1, X_2, \ldots, X_{j-1}$. By suitable construction criteria, convergence to any chosen target distribution is obtained. For MCMC we refer to the literature, for example to [32, 153, 164, 238, 365].

2.7 Exercises

2.1 (Testing a Distribution)

Let X be a random variate with density f and let $a_1 < a_2 < \ldots < a_l$ define a partition of the support of f into subintervals, including the unbounded intervals $x < a_1$ and $x > a_l$. Recall from (B.1), (B.2) that the probability of a realization of X falling into $a_k \le x < a_{k+1}$ is given by

$$p_k := \int_{a_k}^{a_{k+1}} f(x)\,\mathrm{d}x\,,\ k = 1, 2, \ldots, l-1\,,$$

which can be approximated by $(a_{k+1} - a_k) f\left(\frac{a_k + a_{k+1}}{2}\right)$. Perform a sample of j realizations $x_1, \ldots, x_j$ of a random number generator, and denote j_k the number of samples falling into $a_k \le x < a_{k+1}$. For normal variates with density f from (B.9) design an algorithm that performs a simple statistical test of the quality of the $x_1, \ldots, x_j$.

Hints: See Sect. 2.1 for the special case of uniform variates. Argue for what choices of a_1 and a_l the probabilities p_0 and p_l may be neglected. Think about a reasonable relation between l and j.

2.2 (Academic Number Generator)
Consider the random number generator $N_i = 2N_{i-1} \bmod 11$. For $(N_{i-1}, N_i) \in \{0, 1, \ldots, 10\}^2$ and integer tuples with $z_0 + 2z_1 = 0 \bmod 11$ the equation

$$z_0 N_{i-1} + z_1 N_i = 0 \bmod 11$$

defines families of parallel straight lines, on which all points (N_{i-1}, N_i) lie. These straight lines are to be analyzed. For which of the families of parallel straight lines are the gaps maximal?

2.3 (Deficient Random Number Generator)
For some time the generator

$$N_i = aN_{i-1} \bmod M, \quad \text{with } a = 2^{16} + 3,\ M = 2^{31}$$

was in wide use. Show for the sequence $U_i := N_i/M$

$$U_{i+2} - 6U_{i+1} + 9U_i \ \text{ is integer.}$$

What does this imply for the distribution of the triples (U_i, U_{i+1}, U_{i+2}) in the unit cube?

2.4 (Lattice of the Linear Congruential Generator)

(a) Show by induction over j

$$N_{i+j} - N_j = a^j (N_i - N_0) \bmod M$$

(b) Show for integer $z_0, z_1, \ldots, z_{m-1}$

$$\begin{pmatrix} N_i \\ N_{i+1} \\ \vdots \\ N_{i+m-1} \end{pmatrix} - \begin{pmatrix} N_0 \\ N_1 \\ \vdots \\ N_{m-1} \end{pmatrix} = (N_i - N_0) \begin{pmatrix} 1 \\ a \\ \vdots \\ a^{m-1} \end{pmatrix} + M \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_{m-1} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & \cdots & 0 \\ a & M & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a^{m-1} & 0 & \cdots & M \end{pmatrix} \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_{m-1} \end{pmatrix}$$

2.5 (Quality of Fibonacci-Generated Numbers)
Analyze and visualize the planes in the unit cube, on which all points fall that are generated by the Fibonacci recursion

$$U_{i+1} := (U_i + U_{i-1}) \bmod 1 \, .$$

2.6 (Coarse Approximation of Normal Deviates)
Let $U_1, U_2, \ldots$ be independent random numbers $\sim \mathcal{U}[0,1]$, and

$$X_k := \sum_{i=k}^{k+11} U_i - 6 \, .$$

Calculate mean and variance of the X_k.

2.7 (Cauchy-Distributed Random Numbers)
A Cauchy-distributed random variable has the density function

$$f_c(x) := \frac{c}{\pi} \frac{1}{c^2 + x^2} \, .$$

Show that its distribution function F_c and its inverse F_c^{-1} are

$$F_c(x) = \frac{1}{\pi} \arctan \frac{x}{c} + \frac{1}{2} \, , \quad F_c^{-1}(y) = c \tan(\pi(y - \frac{1}{2})) \, .$$

How can this be used to generate Cauchy-distributed random numbers out of uniform deviates?

2.8 (Inversion)
Use the inversion method and uniformly distributed $U \sim \mathcal{U}[0,1]$ to calculate a stochastic variable X with distribution

$$F(x) = 1 - \mathrm{e}^{-2x(x-a)} \, , \; x \geq a \, .$$

2.9 (Laplace Distribution)
The density function of the Laplace distribution is

$$g(x) := \frac{1}{2} \exp(-|x|) \, .$$

(a) Derive the distribution function

$$G(x) := \int_{-\infty}^{x} g(s) \, \mathrm{d}s$$

and its inverse.

(b) Formulate an algorithm that calculates random variates from the G-distribution, applying the inversion method and using $U \sim \mathcal{U}[0,1]$ as input.

2.10 (Inverting the Normal Distribution)
Suppose $F(x)$ is the standard normal distribution function. Construct a rough approximation $G(u)$ to $F^{-1}(u)$ for $0.5 \le u < 1$ as follows:

(a) Construct a rational function $G(u)$ ($\longrightarrow$ Appendix C.1) with correct asymptotic behavior, point symmetry with respect to $(u,x) = (0.5, 0)$, using only one parameter.
(b) Fix the parameter by interpolating a given point $(x_1, F(x_1))$.
(c) What is a simple criterion for the error of the approximation?

2.11 (Time-Changed Wiener Process)
For a time-changing function $\tau(t)$ set $\tau_j := \tau(j\,\Delta t)$ for some time increment Δt.

(a) Argue why Algorithm 1.8 changes to $W_j = W_{j-1} + Z\sqrt{\tau_j - \tau_{j-1}}$ (last line).
(b) Let τ_j be the exponentially distributed jump instances of a Poisson experiment, see Sect. 1.9 and Property 1.20(e). How should the jump intensity λ be chosen such that the expectation of the $\Delta\tau$ is Δt? Implement and test the algorithm, and visualize the results. Experiment with several values of the jump intensity λ.

2.12 (Rejection)
Two density functions g and f are given by

$$f(x) := \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2}) \qquad \text{(Gaussian density)}$$

$$g(x) := \frac{1}{2} \exp(-|x|) \qquad \text{(Laplace density)}$$

Establish the smallest c such that $cg(x) \ge f(x)$ for all $x \in \mathbb{R}$. Apply the rejection method to generate normally distributed x; use Exercise 2.9.

2.13 (Uniform Distribution)
For the uniformly distributed random variables $(V_1,\ V_2)$ on the unit disk consider the transformation

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} V_1^2 + V_2^2 \\ \frac{1}{2\pi} \arg((V_1, V_2)) \end{pmatrix}$$

where $\arg((V_1, V_2))$ denotes the corresponding angle. Show that $(X_1,\ X_2)$ is distributed uniformly.

2.14 (Programming Assignment: Normal Deviates)

(a) Write a computer program that implements the *Fibonacci generator*

$$U_i := U_{i-17} - U_{i-5}$$
$$U_i := U_i + 1 \text{ in case } U_i < 0$$

in the form of Algorithm 2.7.

Tests: Visual inspection of 10,000 points in the unit square.

(b) Write a computer program that implements *Marsaglia's Polar Algorithm* (Algorithm 2.14). Use the uniform deviates from a).

Tests:

1.) For a sample of 5000 points calculate estimates of mean and variance.
2.) For the discretized SDE

$$\Delta x = 0.1\Delta t + Z\sqrt{\Delta t}, \quad Z \sim \mathcal{N}(0,1)$$

calculate some trajectories for $0 \le t \le 1,\ \Delta t = 0.01,\ x_0 = 0$.

2.15 (Ziggurat)

Let f be the normal density function, and (x_i, y_i) for $i = 1, \ldots, N-1$ the coordinates of the vertices of the ziggurat, as indicated in Fig. 2.8, and $y_i := f(x_i)$. (Compare Sect. 2.3.3.) Label the segments $i = 0, \ldots, N-1$ from bottom to top; for $i > 0$ these are rectangular boxes. All segments have equal area A, which is to be determined iteratively.

(a) Assume for a moment the parameter A to be given. Set up an equation that defines x_{N-1} implicitly as function $x_{N-1} = \alpha(A)$.
(b) Set up an equation that defines x_1 implicitly, again depending on A. Then set up a recursion that defines $x_2, \ldots, x_{N-1}$ based on the value x_1. After numerically solving these implicit equations one obtains another version for x_{N-1}, which can be regarded as a function $x_{N-1} = \beta(A)$. Of course both values must be the same, $\alpha(A) = \beta(A)$. This equation can be solved iteratively for A, say, by bisection.
(c) For $N = 8$ formulate an algorithm that calculates A. What is a reasonable initial guess for A? Note that neither α nor β are given explicitly; they can be evaluated numerically.

2.16 (Spectral Decomposition of a Covariance Matrix)

For symmetric positive definite $n \times n$ matrices Σ there exists a set of orthonormal eigenvectors $v^{(1)}, \ldots, v^{(n)}$ and eigenvalues $\lambda_1 \ge \cdots \ge \lambda_n > 0$ such that

$$\Sigma v^{(j)} = \lambda_j v^{(j)}, \quad j = 1, \ldots, n\,.$$

Arrange the n eigenvector columns into the $n \times n$ matrix $B := (v^{(1)}, \dots, v^{(n)})$, and the eigenvalues into the diagonal matrices $\Lambda := \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ and $\Lambda^{\frac{1}{2}} := \operatorname{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n})$.

(a) Show $\Sigma B = B\Lambda$.
(b) Show that

$$A := B\Lambda^{\frac{1}{2}}$$

factorizes Σ in the sense $\Sigma = AA^{tr}$.
(c) Show

$$AZ = \sum_{j=1}^{n} \sqrt{\lambda_j}\, Z_j\, v^{(j)} .$$

(d) And the reversal of Sect. 2.3.4 holds: For a random vector $X \sim \mathcal{N}(0, \Sigma)$ the transformed random vector $A^{-1}X$ has uncorrelated components: Show $\mathsf{Cov}(A^{-1}X) = I$ and $\mathsf{Cov}(B^{-1}X) = \Lambda$.
(e) For the 2×2 matrix

$$\Sigma = \begin{pmatrix} 5 & 1 \\ 1 & 10 \end{pmatrix}$$

calculate the Cholesky decomposition and $B\Lambda^{\frac{1}{2}}$.

Hint: The above is the essence of the *principal component analysis*. Here Σ represents a covariance matrix or a correlation matrix. (For an example see Fig. 2.13.) The matrix B and the eigenvalues in Λ reveal the structure of the data. B defines a linear transformation of the data to a rectangular coordinate system, and the eigenvalues λ_j measure the corresponding variances. In case $\lambda_{k+1} \gg \lambda_k$ for some index k, the sum in (c) can be truncated after the kth term in order to reduce the dimension. The computation of B and Λ (and hence A) is costly, but a dominating λ_1 allows for a simple approximation of $v^{(1)}$ by the power method.

2.17 (Correlated Distributions)
Suppose we need a two-dimensional random variable (X_1, X_2) that must be distributed normally with mean 0, and given variances σ_1^2, σ_2^2 and prespecified correlation ρ. How is X_1, X_2 obtained out of $Z_1, Z_2 \sim \mathcal{N}(0, 1)$?

2.18 (Error of the Monte Carlo Integration)
The domain for integration is $\mathcal{D} = [0, 1]^m$. For

$$\theta_N := \frac{1}{N}\sum_{i=1}^{N} g(x_i)\,, \quad \mathsf{E}(g) := \int g\, \mathrm{d}x\,, \; v := g - \mathsf{E}(g)\,, \; \delta_N := \int g\, \mathrm{d}x - \theta_N$$

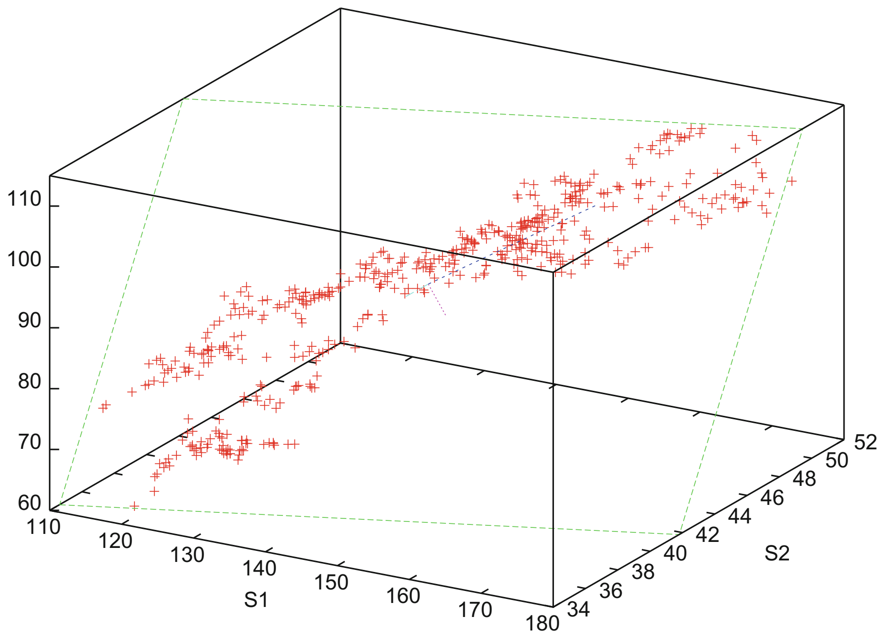

Fig. 2.13 Prices of the DAX assets Allianz (S1), BMW (S2), and HeidelbergCement; 500 trading days from Nov 5, 2005 (in *red*); eigenvalues of the covariance matrix are 400.8, 25.8, 2.73; eigenvectors centered at the mean point and scaled by $\sqrt{\lambda}$ are shown, and the plane (in *green*) spanned by $v^{(1)}$, $v^{(2)}$

and the variance $\sigma^2(g)$ from (2.15) show

(a) $\mathsf{E}(v) = 0$
(b) $\sigma^2(v) = \sigma^2(g)$
(c) $\sigma^2(\delta_N) = \mathsf{E}(\delta_N^2) = \frac{1}{N^2}\int(\sum v(x_i))^2\,\mathrm{d}x = \frac{1}{N}\sigma^2(g)$
Hint on (c): When the random points x_i are i.i.d. (independent identical distributed), then also $g(x_i)$ and $v(x_i)$ are i.i.d. A consequence is $\int v(x_i)v(x_j)\,\mathrm{d}x = 0$ for $i \neq j$.

2.19 (Experiment on Monte Carlo Integration)
To approximate the integral

$$\int_0^1 g(x)\,\mathrm{d}x$$

calculate a Monte Carlo sum

$$\frac{1}{N}\sum_{i=1}^{N} g(x_i)$$

for $g(x) = 5x^4$ and, for example, $N = 100,000$ random numbers $x_i \sim \mathcal{U}[0, 1]$. The absolute error behaves like $cN^{-1/2}$. Compare the approximation with the exact integral for several N and seeds to obtain an estimate of c.

2.20 (Bounds on the Discrepancy)
(Compare Definition 2.16) Show

(a) $0 \le D_N \le 1$,
(b) $D_N^* \le D_N \le 2^m D_N^*$ (show this at least for $m \le 2$),
(c) $D_N^* \ge \frac{1}{2N}$ for $m = 1$.

2.21 (Algorithm for the Radical-Inverse Function)
Use the idea

$$i = \left(d_k b^{k-1} + \ldots + d_1\right) b + d_0$$

to formulate an algorithm that obtains $d_0, d_1, \ldots, d_k$ by repeated division by b. Reformulate $\phi_b(i)$ from Definition 2.18 into the form $\phi_b(i) = z/b^{j+1}$ such that the result is represented as rational number. The numerator z should be calculated in the same loop that establishes the digits $d_0, \ldots, d_k$.