

Chapter 2

An Introduction to Monte Carlo Methods

2.1 What are Monte Carlo Methods?

This lecture course is concerned with Monte Carlo methods, which are sometimes referred to as *stochastic simulation* (Ripley (1987) for example only uses this term).

Examples of Monte Carlo methods include stochastic integration, where we use a simulation-based method to evaluate an integral, Monte Carlo tests, where we resort to simulation in order to compute the p-value, and Markov-Chain Monte Carlo (MCMC), where we construct a Markov chain which (hopefully) converges to the distribution of interest.

A formal definition of Monte Carlo methods was given (amongst others) by Halton (1970). He defined a Monte Carlo method as “representing the solution of a problem as a parameter of a hypothetical population, and using a random sequence of numbers to construct a sample of the population, from which statistical estimates of the parameter can be obtained.”

2.2 Introductory examples

Example 2.1 (A raindrop experiment for computing π). Assume we want to compute an Monte Carlo estimate of π using a simple experiment. Assume that we could produce “uniform rain” on the square $[-1, 1] \times [-1, 1]$, such that the probability of a raindrop falling into a region $\mathcal{R} \subset [-1, 1]^2$ is proportional to the area of \mathcal{R} , but independent of the position of \mathcal{R} . It is easy to see that this is the case iff the two coordinates X, Y are i.i.d. realisations of uniform distributions on the interval $[-1, 1]$ (in short $X, Y \stackrel{\text{i.i.d.}}{\sim} \text{U}[-1, 1]$).

Now consider the probability that a raindrop falls into the unit circle (see figure 2.1). It is

$$\mathbb{P}(\text{drop within circle}) = \frac{\text{area of the unit circle}}{\text{area of the square}} = \frac{\int\int_{\{x^2+y^2 \leq 1\}} 1 \, dx dy}{\int\int_{\{-1 \leq x, y \leq 1\}} 1 \, dx dy} = \frac{\pi}{2 \cdot 2} = \frac{\pi}{4}$$

In other words,

$$\pi = 4 \cdot \mathbb{P}(\text{drop within circle}),$$

i.e. we found a way of expressing the desired quantity π as a function of a probability.

Of course we cannot compute $\mathbb{P}(\text{drop within circle})$ without knowing π , however we can estimate the probability using our raindrop experiment. If we observe n raindrops, then the number of raindrops Z that fall inside the circle is a binomial random variable:

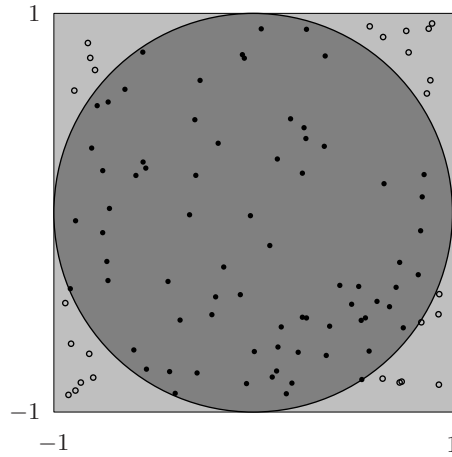


Figure 2.1. Illustration of the raindrop experiment for estimating π

$$Z \sim B(n, p), \quad \text{with } p = \mathbb{P}(\text{drop within circle}).$$

Thus we can estimate p by its maximum-likelihood estimate

$$\hat{p} = \frac{Z}{n},$$

and we can estimate π by

$$\hat{\pi} = 4\hat{p} = 4 \cdot \frac{Z}{n}.$$

Assume we have observed, as in figure 2.1, that 77 of the 100 raindrops were inside the circle. In this case, our estimate of π is

$$\hat{\pi} = \frac{4 \cdot 77}{100} = 3.08,$$

which is relatively poor.

However the *law of large numbers* guarantees that our estimate $\hat{\pi}$ converges almost surely to π . Figure 2.2 shows the estimate obtained after n iterations as a function of n for $n = 1, \dots, 2000$. You can see that the estimate improves as n increases.

We can assess the quality of our estimate by computing a confidence interval for π . As we have $Z \sim B(100, p)$ and $\hat{p} = \frac{Z}{n}$, we use the approximation that $Z \sim N(100p, 100p(1 - p))$. Hence, $\hat{p} \sim N(p, p(1 - p)/100)$, and we can obtain a 95% confidence interval for p using this Normal approximation:

$$\left[0.77 - 1.96 \cdot \sqrt{\frac{0.77 \cdot (1 - 0.77)}{100}}, 0.77 + 1.96 \cdot \sqrt{\frac{0.77 \cdot (1 - 0.77)}{100}} \right] = [0.6875, 0.8525],$$

As our estimate of π is four times the estimate of p , we now also have a confidence interval for π :

$$[2.750, 3.410]$$

In more general, let $\hat{\pi}_n = 4\hat{p}_n$ denote the estimate after having observed n raindrops. A $(1 - 2\alpha)$ confidence interval for p is then

$$\left[\hat{p}_n - z_{1-\alpha} \sqrt{\frac{\hat{p}_n(1 - \hat{p}_n)}{n}}, \hat{p}_n + z_{1-\alpha} \sqrt{\frac{\hat{p}_n(1 - \hat{p}_n)}{n}} \right],$$

thus a $(1 - 2\alpha)$ confidence interval for π is

$$\left[\hat{\pi}_n - z_{1-\alpha} \sqrt{\frac{\hat{\pi}_n(4 - \hat{\pi}_n)}{n}}, \hat{\pi}_n + z_{1-\alpha} \sqrt{\frac{\hat{\pi}_n(4 - \hat{\pi}_n)}{n}} \right]$$

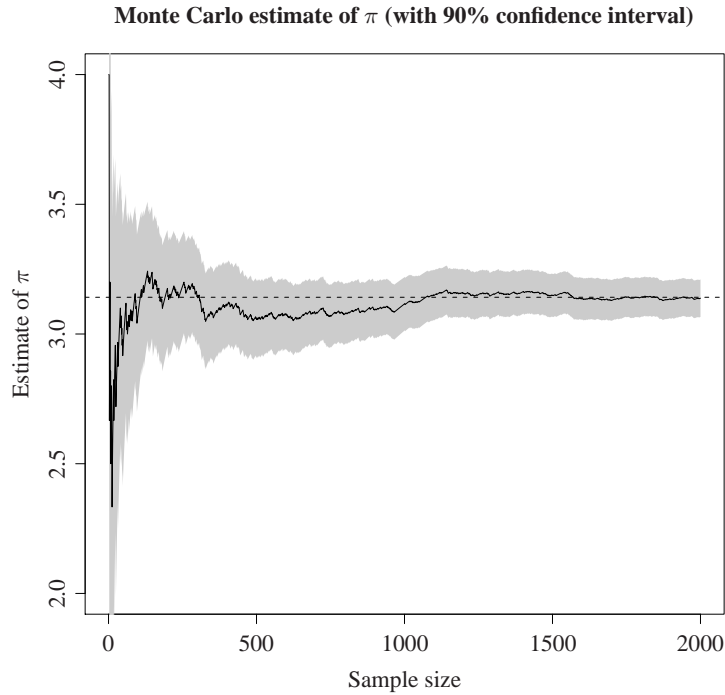


Figure 2.2. Estimate of π resulting from the raindrop experiment

Let us recall again the different steps we have used in the example:

- We have written the quantity of interest (in our case π) as an expectation.¹
- Second, we have replaced this algebraic representation of the quantity of interest by a sample approximation to it. The law of large numbers guaranteed that the sample approximation converges to the algebraic representation, and thus to the quantity of interest. Furthermore we used the central limit theorem to assess the speed of convergence.

It is of course of interest whether the Monte Carlo methods offer more favourable rates of convergence than other numerical methods. We will investigate this in the case of Monte Carlo integration using the following simple example.

Example 2.2 (Monte Carlo Integration). Assume we want to evaluate the integral

$$\int_0^1 f(x) dx \quad \text{with} \quad f(x) = \frac{1}{27} \cdot (-65536x^8 + 262144x^7 - 409600x^6 + 311296x^5 - 114688x^4 + 16384x^3)$$

using a Monte Carlo approach.² Figure 2.3 shows the function for $x \in [0, 1]$. Its graph is fully contained in the unit square $[0, 1]^2$.

Once more, we can resort to a raindrop experiment. Assume we can produce uniform rain on the unit square. The probability that a raindrop falls below the curve is equal to the area below the curve, which of course equals the integral we want to evaluate (the area of the unit square is 1, so we don't need to rescale the result).

A more formal justification for this is, using the fact that $f(x) = \int_0^{f(x)} 1 dt$,

$$\int_0^1 f(x) dx = \int_0^1 \int_0^{f(x)} 1 dt dx = \int \int_{\{(x,t):t \leq f(x)\}} 1 dt dx = \frac{\int \int_{\{(x,t):t \leq f(x)\}} 1 dt dx}{\int \int_{\{0 \leq x,t \leq 1\}} 1 dt dx}$$

The numerator is nothing other than the dark grey area under the curve, and the denominator is the area of the unit square (shaded in light grey in figure 2.3). Thus the expression on the right hand side is the probability that a

¹ A probability is a special case of an expectation as $\mathbb{P}(A) = \mathbb{E}(\mathbb{I}_A)$.

² As f is a polynomial we can obtain the result analytically, it is $\frac{4096}{8505} = \frac{2^{12}}{3^5 \cdot 5 \cdot 7} \approx 0.4816$.

raindrop falls below the curve.

We have thus re-expressed our quantity of interest as a probability in a statistical model. Figure 2.3 shows the result obtained when observing 100 raindrops. 52 of them are below the curve, yielding a Monte-Carlo estimate of the integral of 0.52.

If after n raindrops a proportion \hat{p}_n is found to lie below the curve, a $(1 - 2\alpha)$ confidence interval for the value of the integral is

$$\left[\hat{p}_n - z_{1-\alpha} \sqrt{\frac{\hat{p}_n(1 - \hat{p}_n)}{n}}, \hat{p}_n + z_{1-\alpha} \sqrt{\frac{\hat{p}_n(1 - \hat{p}_n)}{n}} \right]$$

Thus the speed of convergence of our (rather crude) Monte Carlo method is $O_{\mathbb{P}}(n^{-1/2})$. ◁

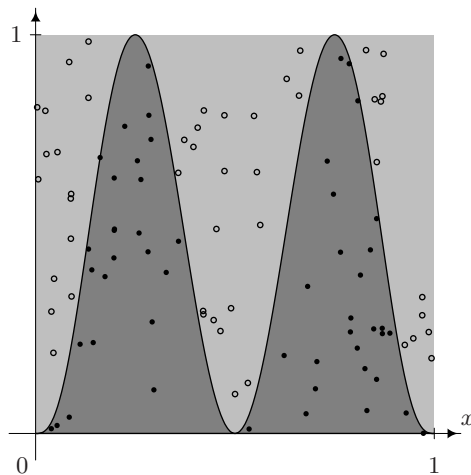


Figure 2.3. Illustration of the raindrop experiment to compute $\int_0^1 f(x)dx$

When using Riemann sums (as in figure 2.4) to approximate the integral from example 2.2 the error is of order $O(n^{-1})$.^{3,4}

Recall that our Monte Carlo method was “only” of order $O_{\mathbb{P}}(n^{-1/2})$. However, it is easy to see that its speed of convergence is of the same order, regardless of the dimension of the support of f . This is not the case for other (deterministic) numerical integration methods. For a two-dimensional function f the error made by the Riemann approximation using n function evaluations is $O(n^{-1/2})$.⁵

This makes the Monte Carlo methods especially suited for high-dimensional problems. Furthermore the Monte Carlo method offers the advantage of being relatively simple and thus easy to implement on a computer.

2.3 A Brief History of Monte Carlo Methods

Experimental Mathematics is an old discipline: the Old Testament (1 Kings vii. 23 and 2 Chronicles iv. 2) contains a rough estimate of π (using the columns of King Solomon’s temple). Monte Carlo methods are a somewhat more recent discipline. One of the first documented Monte Carlo experiments is *Buffon’s needle* experiment (see example 2.3 below). Laplace (1812) suggested that this experiment can be used to approximate π .

³ The error made for each “bar” can be upper bounded by $\frac{\Delta^2}{2} \max |f'(x)|$. Let n denote the number evaluations of f (and thus the number of “bars”). As Δ is proportional to $\frac{1}{n}$, the error made for each bar is $O(n^{-2})$. As there are n “bars”, the total error is $O(n^{-1})$.

⁴ The order of convergence can be improved when using the trapezoid rule and (even more) by using Simpson’s rule.

⁵ Assume we partition both axes into m segments, i.e. we have to evaluate the function $n = m^2$ times. The error made for each “bar” is $O(m^{-3})$ (each of the two sides of the base area of the “bar” is proportional to m^{-1} , so is the upper bound on $|f(x) - f(\xi_{\text{mid}})|$, yielding $O(m^{-3})$). There are in total m^2 bars, so the total error is only $O(m^{-1})$, or equivalently $O(n^{-1/2})$.

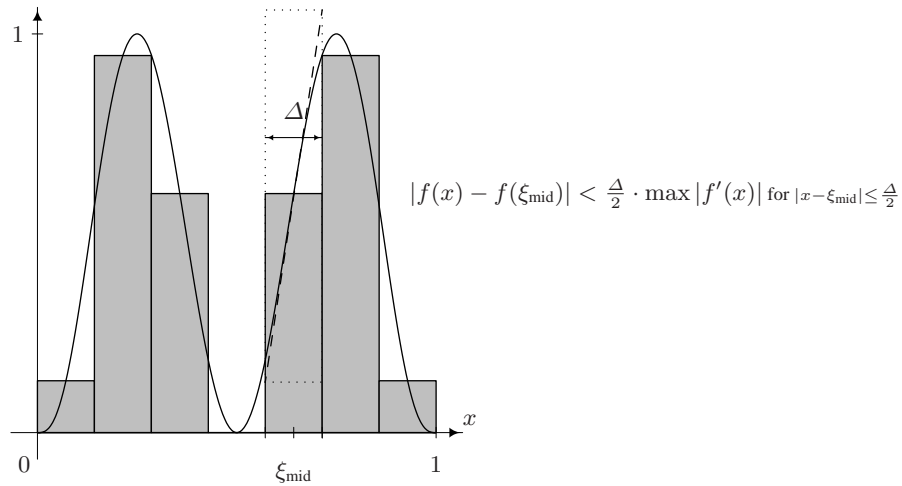


Figure 2.4. Illustration of numerical integration by Riemann sums

Example 2.3 (Buffon’s needle). In 1733, the Comte de Buffon, George Louis Leclerc, asked the following question (Buffon, 1733): Consider a floor with equally spaced lines, a distance δ apart. What is the probability that a needle of length $l < \delta$ dropped on the floor will intersect one of the lines?

Buffon answered the question himself in 1777 (Buffon, 1777).

Assume the needle landed such that its angle is θ (see figure 2.5). Then the question whether the needle intersects a line is equivalent to the question whether a box of width $l \sin \theta$ intersects a line. The probability of this happening is

$$\mathbb{P}(\text{intersect}|\theta) = \frac{l \sin \theta}{\delta}.$$

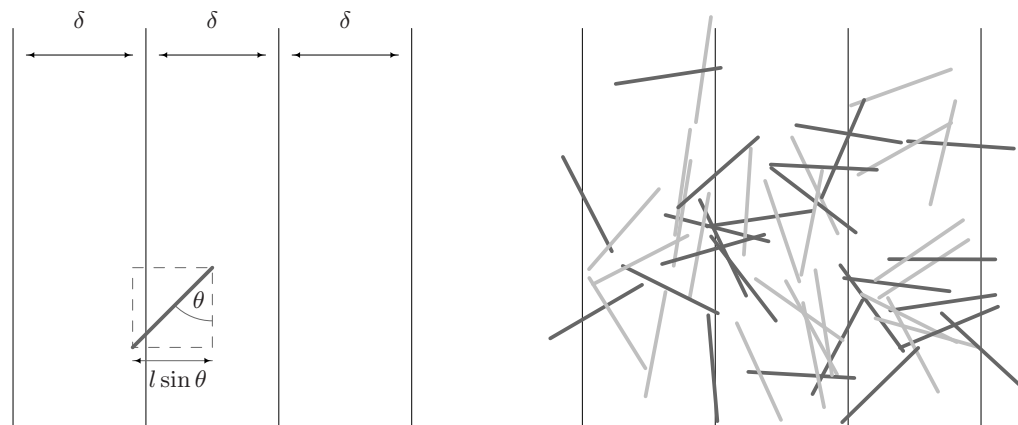
Assuming that the angle θ is uniform on $[0, \pi)$ we obtain

$$\mathbb{P}(\text{intersect}) = \int_0^\pi \mathbb{P}(\text{intersect}|\theta) \cdot \frac{1}{\pi} d\theta = \int_0^\pi \frac{l \sin \theta}{\delta} \cdot \frac{1}{\pi} d\theta = \frac{l}{\pi \delta} \cdot \underbrace{\int_0^\pi \sin \theta d\theta}_{=2} = \frac{2l}{\pi \delta}.$$

When dropping n needles the expected number of needles crossing a line is thus

$$\frac{2nl}{\pi \delta}.$$

Thus we can estimate π by



(a) Illustration of the geometry behind *Buffon’s needle*

(b) Results of the *Buffon’s needle* experiment using 50 needles. Dark needles intersect the thin vertical lines, light needles do not.

Figure 2.5. Illustration of *Buffon’s needle*

$$\pi \approx \frac{2nl}{X\delta},$$

where X is the number of needles crossing a line.

The Italian mathematician Mario Lazzarini performed Buffon's needle experiment in 1901 using a needle of length $l = 2.5\text{cm}$ and lines $d = 3\text{cm}$ apart (Lazzarini, 1901). Of 3408 needles 1808 needles crossed a line, so Lazzarini's estimate of π was

$$\pi \approx \frac{2 \cdot 3408 \cdot 2.5}{1808 \cdot 3} = \frac{17040}{5424} = \frac{355}{133},$$

which is nothing other than the best rational approximation to π with at most 4 digits each in the denominator and the numerator.⁶ ◁

Historically, the main drawback of Monte Carlo methods was that they used to be expensive to carry out. Physical random experiments were difficult to perform and so was the numerical processing of their results.

This however changed fundamentally with the advent of the digital computer. Amongst the first to realise this potential were John von Neuman and Stanisław Ulam, who were then working for the Manhattan project in Los Alamos. They proposed in 1947 to use a computer simulation for solving the problem of neutron diffusion in fissionable material (Metropolis, 1987). Enrico Fermi previously considered using Monte Carlo techniques in the calculation of neutron diffusion, however he proposed to use a mechanical device, the so-called "Fermiac", for generating the randomness. The name "Monte Carlo" goes back to Stanisław Ulam, who claimed to be stimulated by playing poker (Ulam, 1983). In 1949 Metropolis and Ulam published their results in the *Journal of the American Statistical Association* (Metropolis and Ulam, 1949). Nonetheless, in the following 30 years Monte Carlo methods were used and analysed predominantly by physicists, and not by statisticians: it was only in the 1980s — following the paper by Geman and Geman (1984) proposing the Gibbs sampler — that the relevance of Monte Carlo methods in the context of (Bayesian) statistics was fully realised.

2.4 Pseudo-random numbers

For any Monte-Carlo simulation we need to be able to reproduce randomness by a computer algorithm, which, by definition, is deterministic in nature — a philosophical paradox. In the following chapters we will assume that independent (pseudo-)random realisations from a uniform $U[0, 1]$ distribution⁷ are readily available. This section tries to give very brief overview of how pseudo-random numbers can be generated. For a more detailed discussion of pseudo-random number generators see Ripley (1987) or Knuth (1997).

A pseudo-random number generator (RNG) is an algorithm for whose output the $U[0, 1]$ distribution is a suitable model. In other words, the number generated by the pseudo-random number generator should have the same *relevant* statistical properties as independent realisations of a $U[0, 1]$ random variable. Most importantly:

- The numbers generated by the algorithm should reproduce independence, i.e. the numbers X_1, \dots, X_n that we have already generated should not contain any discernible information on the next value X_{n+1} . This property is often referred to as the lack of predictability.
- The numbers generated should be spread out evenly across the interval $[0, 1]$.

In the following we will briefly discuss the linear congruential generator. It is not a particularly powerful generator (so we discourage you from using it in practise), however it is easy enough to allow some insight into how pseudo-random number generators work.

⁶ That Lazzarini's experiment was that precise, however, casts some doubt over the results of his experiments (see Badger, 1994, for a more detailed discussion).

⁷ We will only use the $U(0, 1)$ distribution as a source of randomness. Samples from other distributions can be derived from realisations of $U(0, 1)$ random variables using deterministic algorithms.

Algorithm 2.1 (Congruential pseudo-random number generator). 1. Choose $a, M \in \mathbb{N}$, $c \in \mathbb{N}_0$, and the initial value (“seed”) $Z_0 \in \{1, \dots, M - 1\}$.

2. For $i = 1, 2, \dots$

$$\text{Set } Z_i = (aZ_{i-1} + c) \bmod M, \text{ and } X_i = Z_i/M.$$

The integers Z_i generated by the algorithm are from the set $\{0, 1, \dots, M - 1\}$ and thus the X_i are in the interval $[0, 1)$.

It is easy to see that the sequence of pseudo-random numbers only depends on the seed Z_0 . Running the pseudo-random number generator twice with the same seed thus generates exactly the same sequence of pseudo-random numbers. This can be a very useful feature when debugging your own code.

Example 2.4. Consider the choice of $a = 81$, $c = 35$, $M = 256$, and seed $Z_0 = 4$.

$$\begin{aligned} Z_1 &= (81 \cdot 4 + 35) \bmod 256 = 359 \bmod 256 = 103 \\ Z_2 &= (81 \cdot 103 + 35) \bmod 256 = 8378 \bmod 256 = 186 \\ Z_3 &= (81 \cdot 186 + 35) \bmod 256 = 15101 \bmod 256 = 253 \\ &\dots \end{aligned}$$

The corresponding X_i are $X_1 = 103/256 = 0.4023438$, $X_2 = 186/256 = 0.72656250$, $X_3 = 253/256 = 0.98828120$. ◁

The main flaw of the congruential generator is its “crystalline” nature (Marsaglia, 1968). If the sequence of generated values X_1, X_2, \dots is viewed as points in an n -dimension cube⁸, they lie on a finite, and often very small number of parallel hyperplanes. Or as Marsaglia (1968) put it: “the points [generated by a congruential generator] are about as randomly spaced in the unit n -cube as the atoms in a perfect crystal at absolute zero.” The number of hyperplanes depends on the choice of a , c , and M .

An example for a notoriously poor design of a congruential pseudo-random number generator is RANDU, which was (unfortunately) very popular in the 1970s and used for example in IBM’s System/360 and System/370, and Digital’s PDP-11. It used $a = 2^{16} + 3$, $c = 0$, and $M = 2^{31}$. The numbers generated by RANDU lie on only 15 hyperplanes in the 3-dimensional unit cube (see figure 2.6).

Figure 2.7 shows another cautionary example (taken from Ripley, 1987). The left-hand panel shows a plot of 1,000 realisations of a congruential generator with $a = 1229$, $c = 1$, and $M = 2^{11}$. The random numbers lie on only 5 hyperplanes in the unit square. The right hand panel shows the outcome of the Box-Muller method for transforming two uniform pseudo-random numbers into a pair of Gaussians (see example 3.2).

Due to this flaw of the congruential pseudo-random number generator, it should not be used in Monte Carlo experiments. For more powerful pseudo-random number generators see e.g. Marsaglia and Zaman (1991) or Matsumoto and Nishimura (1998). GNU R (and other environments) provide you with a large choice of powerful random number generators, see the corresponding help page (`?RNGkind`) for details.

⁸ The $(k + 1)$ -th point has the coordinates $(X_{nk+1}, \dots, X_{nk+n-1})$.

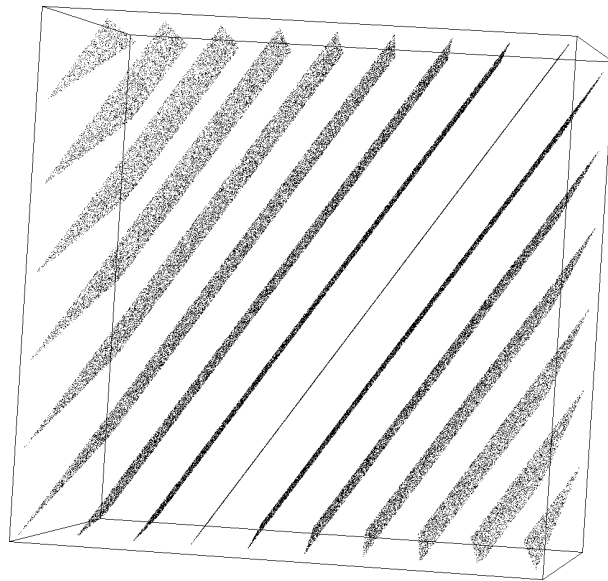
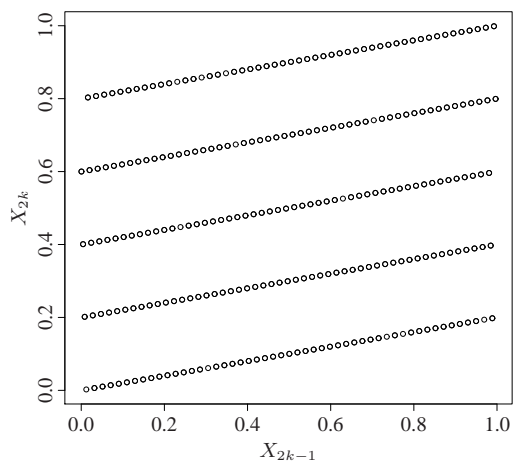
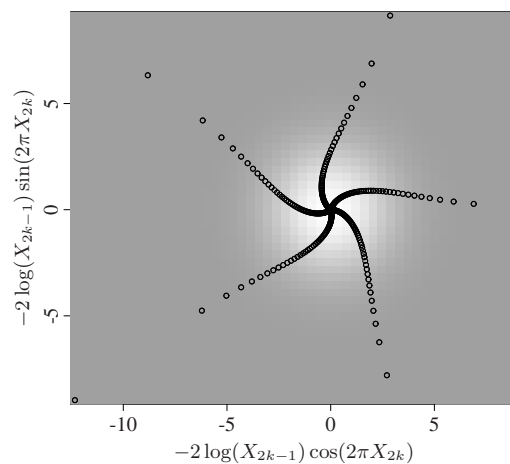


Figure 2.6. 300,000 realisations of the RANDU pseudo-random number generator plotted in 3D. A point corresponds to a triplet $(x_{3k-2}, x_{3k-1}, x_{3k})$ for $k = 1, \dots, 100000$. The data points lie on 15 hyperplanes.



(a) 1,000 realisations of this congruential generator plotted in 2D.



(b) Supposedly bivariate Gaussian pseudo-random numbers obtained using the pseudo-random numbers shown in panel (a).

Figure 2.7. Results obtained using a congruential generator with $a = 1229$, $c = 1$, and $M = 2^{11}$