13 Monte Carlo Methods

Modern science deals with increasingly challenging and complex systems. While relatively simple mathematical models can often be formulated, their implementation for the description of real-world systems is typically not amenable to analytical solutions. For instance, one may wish to "evolve" the state of a system (e.g., in a microscopic model of nucleus-nucleus collisions) with time to study its many configurations or one may want to calculate the acceptance of a large detector for the study of particles produced by elementary particle collisions (proton–proton or heavy ion collisions at the Large Hadron Collider), and so on. The number of states, configuration, or variables may, however, be prohibitively large, or the boundary conditions of the problem very complex, so it is not possible to sample or integrate the full configuration of the phase space of a system by conventional integration techniques. In these instances, it is possible to resort to Monte Carlo methods to sample the many possible states of the system or to obtain the integral of arbitrary functions over the system's configuration or phase space.

We begin our description of Monte Carlo methods with a short discussion of the principles of the method in §13.1. We then show, in §13.2, that Monte Carlo simulations essentially amount to a form of integration that can be most effectively used to compute multidimensional integrals. The notion of pseudorandom numbers and basic techniques commonly used for the generation of pseudorandom numbers are then presented in §13.3. Selected examples of basic applications are introduced in §13.4. More elaborate techniques for applications in nuclear sciences are presented in Chapter 14.

13.1 Basic Principle of Monte Carlo Methods

Monte Carlo methods use "chance" or more properly said random numbers to sample all possible states of a system. The name *Monte Carlo* is inspired from games of chance played at the famous Monte Carlo casinos in Monaco, located south of France. The concept and first implementation of the method originates from Los Alamos National Laboratory, New Mexico, where mathematicians John von Neumann and Stanislaw Ulam, working on the Manhattan Project in the 1940s, proposed carrying out theoretical calculations of radiation shielding using experimental modeling on a computer based on chance. Given the secrecy of the project, von Neuman allegedly chose the name "Monte Carlo" in reference to the casinos where Ulam's uncle would borrow money to gamble.

Random methods of calculation, or stochastic simulations, may be traced back to the pioneers of probability theory, but more specifically to the preelectronic era. Monte Carlo

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

simulations were essential for the development of the Manhattan Project, though, severely limited by the computational tools available at the time. It was only with the development of increasingly powerful computers that Monte Carlo methods were systematically explored and increasingly used for military purposes and later in physical sciences and operations research. The Rand Corporation and the U.S. Air Force were two of the major organizations responsible for funding and disseminating information on Monte Carlo methods during this time, and they began to find a wide application in many different fields. In fact, one of the early pseudorandom number generator function, rand, is named after the Rand Corporation.

Simulations, modeling, or Monte Carlo calculations require a reliable set of random numbers. Truly random numbers might in principle be obtained by observation of stochastic phenomena. One could, for instance, utilize the time interval between decays of radio isotopes or the time interval between the passages of cosmic rays through a particle detector. One could also use the electronic noise intrinsically present in any electronic circuit. By their nature, stochastic phenomena are such that given one "event" it is impossible to predict when the next will occur. Additionally, one does not expect correlations between the occurrence of the events; a sequence of events is intrinsically unpredictable. The generation of random numbers in this fashion would require a device to measure the natural phenomenon (e.g., electronic noise) and digitization of the signal. Both are often impractical and even problematic. For instance, use of electronic noise would require white noise, for example, signals whose Fourier transform yields a spectrum where all frequencies have equal probability (uniform probability) density over a wide range of frequencies. The digitization process is also challenging. A useful random generation must produce numbers with a fine granularity. It should have the capacity to produce billions of different numbers in order to permit simulation with high resolution and avoid coarse binning of the observables or variables being simulated. This would in practice require digitization with an exceeding large number of bits to enable production of random number sets with fine granularity that provide a reasonable approximation of a subset of the set of real numbers, $\mathbb{R}$. Although these requirements can be met in principle, their realization is impractical for most computing applications, and generally considered not worth doing. Mathematicians, scientists, and engineers alike thus commonly use pseudorandom numbers generated algorithmically by a computer program. This said, note that there exists providers of true random numbers based, for instance, on atmospheric noise [100].

The production and study of pseudorandom numbers is an area of research of its own. We here focus on the essential concepts commonly used in the generation of random numbers but include references to more advanced or specialized works.

13.2 Monte Carlo Integration

It is convenient to first introduce the Monte Carlo method as a technique to carry out integrals of single and multidimensional functions. This is of particular interest because, in essence, all applications of the Monte Carlo technique may be reduced to the calculation of some integral.

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

In order to motivate the Monte Carlo integration method, we first review a few theorems of basic calculus (without demonstration).

Mean Value Theorem for Integrals

If a function, $f(x)$, is continuous over a closed interval $[a, b]$, then there exists a number c, with $a < c < b$, such that

$$f(c) = \frac{1}{b-a}\int_a^b f(x)\,dx. \tag{13.1}$$

Composite Midpoint Rule

The area under a curve, $f(x)$, in a given interval $[a, b]$, can be calculated by partitioning the interval in n subintervals $\{[x_{k-1}, x_k]\}_{k=1}^n$ of equal width, h. As per the mean value theorem stated previously, there are values c'_k for each subinterval such that

$$\int_{x_{k-1}}^{x_k} f(x)\,dx = (x_k - x_{k-1})\,f(c'_k) = hf(c_k{}'). \tag{13.2}$$

The integral of $f(x)$ in the interval $[a, b]$ may then be written

$$\int_a^b f(x)\,dx = \sum_{k=1}^n \int_{x_{k-1}}^{x_k} f(x)\,dx = h\sum_{k=1}^n f(c'_k). \tag{13.3}$$

If instead of using the values c'_k satisfying Eq. (13.2), one uses the midpoints $c_k = a + (k - \frac{1}{2})h$, for $k = 1, 2, \ldots, n$, then one can write:

$$\int_a^b f(x)\,dx = M(f, h) + E_M(f, h), \tag{13.4}$$

where

$$M(f, h) = \frac{b-a}{n}\sum_{k=1}^n f(c_k) \tag{13.5}$$

is known as the **composite midpoint rule** and constitutes an approximation of the integral in Eq. (13.4) with error $E_M(f, h)$. One can show that there exists a value c with $a < c < b$ such that the error $E_M(f, h)$ is given by

$$E_M(f, h) = \frac{1}{24}(b-a)\,h^2 f^2(c). \tag{13.6}$$

As such, the error $E_M(f, h)$ is proportional to the square of the bin size h while the area is linear in h. This implies it is possible to obtain an arbitrarily suitable approximation of the integral by choosing a sufficiently small value of the bin width h.

Monte Carlo Integration Rule

The Monte Carlo method is based on the composite midpoint rule approximation. However, rather than evaluating the functions at the midpoint of bins of equal width, one chooses n randomly distributed points x_1, x_2, x_n in the interval $[a, b]$ and evaluates the

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

average of the function at these points:

$$\hat{f} = \frac{1}{n}\sum_{i=1}^{n} f(x_i). \tag{13.7}$$

An approximation of the integral is then obtained by multiplying this average by the width of the interval $[a, b]$:

$$\int_a^b f(x)\,dx = (b-a)\hat{f} + E_{MC}(f, n). \tag{13.8}$$

The error on the integral, $E_{\mathrm{MC}}(f, n)$, can be estimated from

$$E_{\mathrm{MC}}(f, n) = (b-a)\sqrt{\frac{\widehat{f^2} - (\hat{f})^2}{n}}, \tag{13.9}$$

where

$$\widehat{f^2} = \frac{1}{n}\sum_{i=1}^{n} f^2(x_i). \tag{13.10}$$

The error is thus proportional to the variance of $f(x_i)$ and inversely proportional to the square root of the number of points (samples) included in the MC calculation. As such, one concludes that the estimate converges slowly to zero with $n \to \infty$ and is of order $n^{-1/2}$ for finite n.

For single-variable functions, the Monte Carlo integration converges slower than conventional techniques such as the trapezoid method, which converges as n^{-2}, or the m-point Gauss method, which converges as n^{-2m+1}. A numerical integral of a single variable function is thus most effectively obtained with the Gauss method. But integration over multidimensional space using the trapezoid or Gauss methods (or similar methods) can quickly become prohibitive. For instance, a system with m particles has as many as $3 \times m$ degrees of freedom (excluding rotational degrees of freedom): integration using say 100 bins in each dimension would require $100^{3\times m}$ bins and is thus clearly impractical for large values of m, even with modern computers. However, one can show that the convergence of a conventional integration technique in d dimensions scales as $n^{-2/d}$ and is thus slower than the MC convergence ($n^{-1/2}$) for problems involving $d \geq 5$ dimensions. The use of Monte Carlo techniques for problems of five or more dimensions (degrees of freedom) is thus clearly advantageous and commonly used in practical applications.

13.3 Pseudorandom Number Generation

13.3.1 Definition and Basic Criteria

It is usually preferable to use pseudorandom numbers rather than attempting the production of true random numbers. Pseudorandom numbers are generated by use of computer algorithms designed to produce a sequence of number of apparently uncorrelated numbers

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

that are uniformly distributed over a predetermined range. Modern algorithms provide very long sequences of random numbers that fulfill typical requirements of simulations carried out in physics as well as in the simulation of engineered systems.[1] Pseudorandom generators also provide the possibility of repeating the same exact experiment (simulation) many times, thereby making debugging of complex simulation programs considerably easier. In addition, modern random number generators are portable, thereby enabling the production of identical number sequences on different computers and even different operating systems, compilers, and computer architectures. A program can thus be used and tested on different computers with predictably identical results.

Random number generators are typically required to meet the following basic criteria:

1. They must be uniformly distributed within a specific range.
2. They must satisfy statistical tests of randomness, lack predictability, and there should be no correlations between neighboring numbers of a sequence.
3. An algorithm should produce a large sequence of different numbers before repeating a cycle.
4. The calculation should be very fast.

Let us consider each of these points in some detail. (1) Uniformly distributed numbers are required to ensure no region of the parameter space used in a simulation is more probable than others. (2) Randomness and unpredictability are obviously required as the basis of Monte Carlo simulations. Absence of correlations between nearby numbers of a sequence is required to ensure there are no unintended autocorrelations or intercorrelations between the generated simulation variables. (3) Given their algorithmic nature, pseudorandom numbers based on a finite number of bits architecture may feature two undesirable traits: Poor designs lead to algorithms getting stuck on a specific value, for instance zero. The presence of such a fixed end-point implies a sequence is usable only for a finite number of calls to the generator. Good designs feature sequences that do not get stuck on a specific value but eventually repeat: they are cyclical with a finite period. Obviously, cycles should be as long as possible to ensure truly uniform and nonclumpy distributions. (4) Lastly, speed is required to enable rapid and efficient simulations where the CPU's time is spent on the model calculations rather than the generation of random numbers.

13.3.2 Production of Uniform Deviates

There are numerous random number generators on the market (for a relatively recent review, see [114]). These may be included as part of software packages such as Mathematica®, MATLAB®, and ROOT, or as standalone codes provided with compiler libraries. Source codes are available from many vendors as well.

In essence, all generators use the same fundamental concept based on a multiplication method called **uniform deviates method**. The method requires an integer r_0 be chosen as a seed, which provides a starting value, and two integer constants a and m. Successive

[1] Actually, Monte Carlo simulations are now commonly used in all sciences.

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

pseudorandom numbers r_i are produced sequentially using the recursion relation

$$r_{i+1} = (a \times r_i) \mod m, \tag{13.11}$$

where mod stands for the "modulo" operation that yields the remainder of the left number division by the integer m. Studies have demonstrated that with a careful choice of the constants a and m, one can obtain finite sequences of numbers that appear to be randomly distributed between 1 and $m - 1$. It thus suffices to divide the numbers by m to obtain seemingly continuous random numbers in the range $[0, 1[$. The length of a sequence is determined by the choice of the constants and is limited by the computer word size, that is, the number of bits used in integer computation. For instance, values $m = 37$, and $a = 5$ lead to a sequence of 36 "random" numbers. Obviously, practical implementations involve much larger constant values, which produce much longer sequences. There are also several variations of this multiplication algorithm [114].

Detailed statistical studies must be made to ensure that untested random generators produce acceptable sequences of numbers. Failure to use generators abiding by the criteria listed in the previous section may lead to erroneous results and conclusions. In the 1970s, a number of commercial generators provided with FORTRAN compilers and operating systems featured unsuitable degrees of correlations. However, in recent years, better care has been taken by suppliers, and commercially available generators are typically quite reliable.

The numbers produced by the multiplication algorithm, Eq. (13.11), are not truly random, and one might be concerned about the existence of hidden correlations. A basic technique used to improve the generated sequences is to shuffle the numbers of a sequence. In effect, one uses two sequences of numbers simultaneously with distinct values of constant a and m: one sequence is stored in an array and a number from the second sequence is used as an index to select numbers from the first sequence. Obviously, this technique is limited by storage and capacity. Local shuffling within a block of numbers can be also used.

The ROOT environment provides four distinct uniform random generators named TRandom, and TRandom$X$, with $X = 1, 2, 3$. The TRandom3, based on the "Mersenne Twister generator" [141], features a period of about 10^{6000}, is quite fast, and is as such most likely the best choice for the generation of uniform deviates. Note that the availability of several distinct generators makes it possible to carry out calculations with more than one generator and verify that the results are reasonably generator independent.

A discussion of techniques to evaluate the degree of correlation produced by a given generator is beyond the scope of this textbook but may be found, for instance, in ref. [189].

13.3.3 Inversion (Transformation) Method

Consider the problem of generating random numbers according to a nonnegative function, $f(x)$, in some allowed range $x_{\min} \leq x < x_{\max}$. We wish to produce numbers x randomly in such a way that the probability of such generated numbers to be within the interval $[x, x + dx]$ is proportional to $f(x)\,dx$. The function $f(x)$ does not need be normalized so that its integral over the chosen applicability range equals unity. In fact, as discussed in the previous section, a Monte Carlo technique can actually be used to evaluate that integral.

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

The **inversion method** is applicable to a function $f(x)$ whenever it is possible to find a primitive function $F(x)$ and calculate its inverse $F^{-1}(x)$. Given that the goal is to produce random numbers in the range $x_{\min} \leq x < x_{\max}$ with probability $f(x)\,dx$, consider the integral

$$\int_{x_{\min}}^{x} f(x')\,dx' = \int_{x_{\min}}^{x} dF(x') = F(x) - F(x_{\min}) \tag{13.12}$$

The weight associated to the integral in dF is unity. Effectively, this corresponds to having uniformly distributed random numbers, that is, the distribution of F is uniformly distributed. One can then use a uniform random generator to select a value of F and use its inverse F^{-1} to determine the corresponding value x. The integral of $f(x)$ over the range $x_{\min} \leq x' < x$ is a linear fraction R of the integral over the range $x_{\min} \leq x' < x_{\max}$ which one may write

$$\int_{x_{\min}}^{x} f(x')dx' = F(x) - F(x_{\min}) = R\left(F(x_{\max}) - F(x_{\min})\right). \tag{13.13}$$

Solving the preceding for $F(x)$,

$$F(x) = R\left(F(x_{\max}) - F(x_{\min})\right) + F(x_{\min}), \tag{13.14}$$

and inverting the function to obtain the value of x, one gets

$$x = F^{-1}\left(R\left(F(x_{\max}) - F(x_{\min})\right) + F(x_{\min})\right). \tag{13.15}$$

The random generation of numbers distributed according to a function $f(x)$ is thus relatively straightforward and may be obtained according to the following basic algorithm. First, use a uniform random generator bound between 0 and 1 to determine a value of R, and next insert this value in Eq. (13.15) to produce a value of x randomly distributed according to $f(x)$.

Before we proceed with the implementation of the inversion method in two examples below, note that, although simple and efficient, the method is unfortunately applicable only in a very limited number of cases: those for which the function $f(x)$ has an analytical primitive (i.e., indefinite integral) $F(x)$ easily invertible to provide an analytical solution. If the function $F(x)$ exists but cannot be inverted, Eq. (13.15) cannot be applied directly, but it is nonetheless possible to obtain a generator by using a numerical inversion technique. For cases where no primitive function $F(x)$ exists, one must consider alternative approaches, such as those presented in §§13.3.4 and 13.3.5.

Example 1: Linearly Distributed Random Numbers

Problem

Build a random generator that produces numbers linearly distributed, that is, according to $f(x) = ax$, in an arbitrary range $[x_{\min}, x_{\max}]$, with $x_{\max} > 0$.

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

Solution

The function $f(x)$ is integrable, which means it is possible to use the inversion method. The integral of $f(x)$ is

$$y = F(x) = \frac{1}{2}ax^2. \tag{13.16}$$

Given a value y, one solves for x according to

$$x = \sqrt{\frac{2y}{a}}. \tag{13.17}$$

The inverse function F^{-1} may then be written

$$F^{-1}(y) = \sqrt{\frac{2y}{a}}. \tag{13.18}$$

Now insert this expression into Eq. (13.15) to obtain a linearly distributed value of x:

$$x = \sqrt{Rx_{\max}^2 + (1-R)x_{\min}^2}. \tag{13.19}$$

Example 2: Exponentially Distributed Random Numbers

Problem

Build a random generator that produces numbers distributed according to an exponential distribution:

$$f(x) = ke^{-x/\lambda}. \tag{13.20}$$

Solution

The indefinite integral of the preceding exponential is $F(x) = -k\lambda e^{-x/\lambda}$. Its inverse is

$$F^{-1}(y) = -\lambda \ln\left(-\frac{y}{k\lambda}\right). \tag{13.21}$$

Application of Eq. (13.15) yields

$$x = -\lambda \ln\left(R\exp(-x_{\max}/\lambda) + (1-R)\exp(-x_{\min}/\lambda)\right), \tag{13.22}$$

where R is a uniformly distributed random number in the range [0, 1].

13.3.4 Acceptance/Rejection Method

Often times, the transformation method is not applicable because the analytical integral of a function does not exist, or because, if it does exist, it is impossible to invert it. One must then explore alternative methods. We first consider the **acceptance/rejection method**, which is widely applicable provided the maximum $f_{\max}$ of the function $f(x)$ is known, that is, if the function satisfies $f(x) \leq f_{\max}$ in the range of interest.

To apply this method, consider an arbitrary value of x in the range $[x_{\min}, x_{\max}]$ has been selected. The quantity $f(x)dx$ is proportional to the probability of having this particular

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

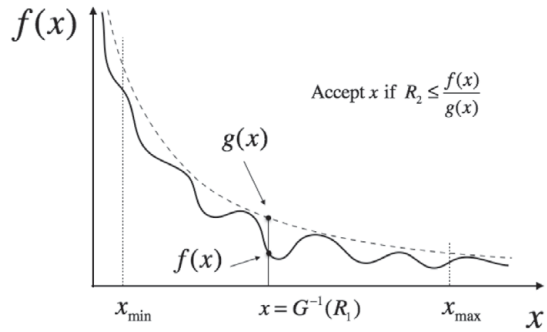


Fig. 13.1 Schematic illustration of the principle of the inversion + acceptance hybrid method.

value of x. Given the function maximum $f_{\max}$ is known, one can express this probability as $f(x)/f_{\max}$, and one can then use a uniformly generated random number in the range [0, 1] to decide whether the selected value of x is accepted, that is, one accepts the value of x provided the random number R is smaller than the relative probability $f(x)/f_{\max}$.

The acceptance/rejection method requires the use of two random numbers: the first is used to select a specific value of x and the second to determine whether this value is accepted. It may then be implemented according to the following algorithm:

1. Generate two uniformly distributed random numbers r_1 and r_2 in the range [0, 1].
2. Use the first number r_1 to select a value x with a uniform probability in the allowed range $[x_{\min}, x_{\max}]$:

$$x = x_{\min} + r_1 \, (x_{\max} - x_{\min}) \, . \tag{13.23}$$

3. Compare r_2 with the ratio $f(x)/f_{\max}$; if $r_2 > f(x)/f_{\max}$, and then reject this instance of x and start over at step 1 for a new try. Otherwise retain the value of x as the final answer.

The acceptance/rejection method works for any function $f(x)$ with an upper-bound $f_{\max}$, but it may become highly inefficient (and slow) if the function $f(x)$ has large spikes or varies dramatically over the range of interest $[x_{\min}, x_{\max}]$. The efficiency of a pseudorandom generator method is defined as the probability that a value of x will be retained at any given try. In the context of the acceptance/rejection method, it amounts to the ratio of the integral of the function over the range $[x_{\min}, x_{\max}]$ by the area $f_{\max}(x_{\max} - x_{\min})$. The efficiency can thus be low if the integral of $f(x)$ is much smaller than the area, that is, if $f(x)$ is in general much smaller than $f_{\max}$.

There are also cases when an upper bound cannot be defined. This is the case, for instance, for functions of the form $1/x^{\alpha}$, with $\alpha > 0$, evaluated with a lower bound at zero. Variable transformations may then be used to make the function smoother. For instance, using $y = \ln x$ would enable a roughly constant function over a narrow interval and therefore yield a fairly large efficiency.

In view of the limited efficiency of the acceptance/rejection method, it is often convenient to combine it with the inversion method discussed in the previous section. This is accomplished, as illustrated in Figure 13.1, by using an *envelope* function $g(x)$, which satisfies the two following conditions: (1) The value of the function $g(x)$ is such that

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

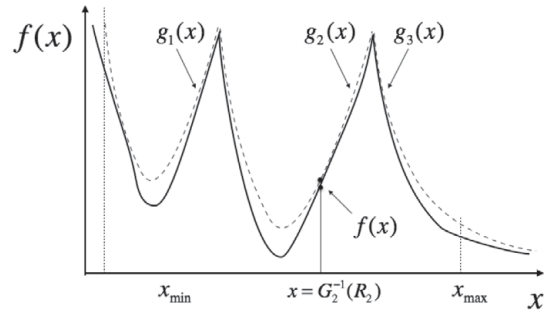


Fig. 13.2 Schematic illustration of the principle of the composite inversion + acceptance hybrid method.

$f(x) \leq g(x)$ everywhere over the interval of interest and (2) the function $g(x)$ must have an easily calculable primitive function $G(x)$, which is invertible, that is, with a known analytical solution for $G^{-1}(y)$. A trial value x is generated according to $x = G^{-1}(R_1)$, with uniform deviate $R_1 \in [0, 1]$, and a second uniform deviate $R_2 \in [0, 1]$ is used to decide whether the selected value of x shall be accepted and returned. The value is accepted if R_2 is smaller than the ratio $f(x)/g(x)$. This hybrid method operates similarly as the acceptance/rejection method but offers better efficiency because rather than choosing x values uniformly over the interval of interest, one uses a function $g(x)$ to **presample** x. Ideally, a function $g(x)$ mimics the overall behavior of the function $f(x)$, thereby providing an increased efficiency. The efficiency of the method is determined by the ratio of the integral of the two functions over the interval. And, provided $g(x) < f_{\max}$ over the interval $[x_{\min}, x_{\max}]$ of interest, one finds the efficiency of this method is greater than that of the basic acceptance/rejection method. The efficiency of the method is

$$\epsilon = \frac{\int_{x_{\min}}^{x_{\max}} f(x)\,dx}{\int_{x_{\min}}^{x_{\max}} g(x)\,dx} > \frac{\int_{x_{\min}}^{x_{\max}} f(x)\,dx}{f_{\max}(x_{\max} - x_{\min})}. \tag{13.24}$$

The algorithm of this hybrid method can be summarized as follows:

1. Generate a random number R_1 with uniform distribution in the range [0, 1], and use the inverse function $G^{-1}(R_1)$ to obtain a trial version of x.
2. Generate a random number R_2 with uniform distribution in the range [0, 1].
3. If $R_2 > f(x)/g(x)$, reject this value of x value and start over. Otherwise retain the value of x as the final answer.

The method first selects x according to the probability $g(x)\,dx = dG(x)$. Then the acceptance/rejection part of the algorithm accepts x with a probability $f(x)/g(x)$. The overall probability of getting a specific value of x is thus $g(x)\,dx \times f(x)/g(x) = f(x)\,dx$ as needed to properly sample the function $f(x)$.

Functions used in theoretical calculations often feature several spikes, as illustrated in Figure 13.2. It may then be difficult to find a "simple" function with similar behavior and each spike may have to be handled separately. This is achieved by partitioning the range of interest in n subdomains where one can apply n sampling functions $g_k(x)$, with $k = 1, \ldots, n$. Clearly, each of the functions must satisfy the condition $f(x) \leq g_k(x)$ in their

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

respective range of validity and have an invertible primitive function $G_k^{-1}(y)$. One can thus define a composite function $g(x)$ as

$$g(x) = \begin{cases} g_1(x) \text{ for } x_{\min} \leq x < x_1 \\ \cdots \quad \cdots \\ g_k(x) \text{ for } x_{k-1} \leq x < x_k \\ \cdots \quad \cdots \\ g_n(x) \text{ for } x_{n-1} \leq x < x_{\max} \end{cases} \tag{13.25}$$

such that the functions $g_k(x)$ have primitive functions, and can be inverted. Generation of pseudorandom numbers may then be achieved with the following algorithm:

1. Generate a uniform deviate $R_1 \in [0, 1[$ to select an integer value k satisfying $P_{k-1} \leq R_1 < P_K$ at random in the range $[1, n]$, based on the cumulative probabilities $P_k = \sum_{i=1}^{k} p_i$ for $1 \leq k \leq n$ and $P_0 = 0$, where the coefficients p_i represent integrals of the functions $g_i(x)$ in their respective ranges of applicability
$$p_i = \int_{x_{min,i}}^{x_{max,i}} g_i(x)\, dx. \tag{13.26}$$
2. Given the function g_k selected, generate a random number R_2 with uniform distribution in the range $[0, 1[$ and use the inverse method to generate a value x according to
$$x = G_k^{-1}\left(G_k(x_{\min,\mathrm{k}}) + R_2\left[G_k(x_{\max,\mathrm{k}}) - G_k(x_{\min,\mathrm{k}})\right]\right) \tag{13.27}$$
3. Generate a random number R_3 and compare it to the ratio $f(x)/g_k(x)$; if larger, then reject the value and return to step 1 for a new try. Otherwise, retain and use the most recent value of x as the final answer.

13.3.5 Integrated Histogram Method

The **Integrated Histogram method** is a numerical technique based on the inversion method discussed in §13.3.3. It is commonly used either as a practical technique to handle non-integrable functions or for the generation of random numbers based on histograms measured experimentally, particularly in the context of analyses of the performance of a measurement or detection system.

Recall that the inversion method requires inversion of the indefinite integral, $F(x)$, of a function $f(x)$ one wishes to sample. The integrated histogram method replaces the function $f(x)$ by a histogram $h(x)$ meant to be a reasonable representation or approximation of the function. Additionally, it replaces the analytical inversion of the function $F(x)$ by a numerical inversion based on a binary search across H_x, a histogram consisting of a running sum of h_x, as schematically illustrated in Figure 13.3. Let us examine the method in more detail.

Consider the generation of random numbers in the range $[x_{\min}, x_{\max}]$ of interest. The basic idea of the method is to replace the function $f(x)$ by a histogram h_x with sufficiently many bins to enable a reasonable representation of the function. If $f(x)$ is known exactly, one has the latitude to decide the precision with which the histogram must be binned.

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

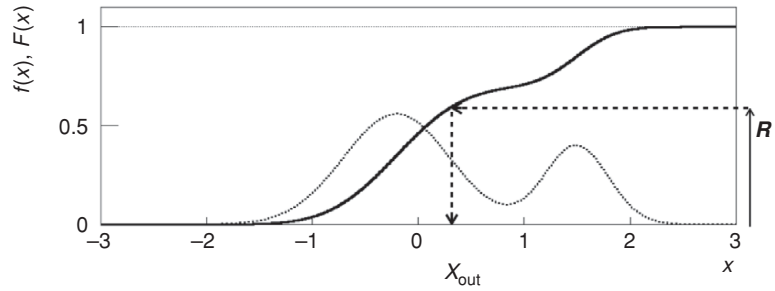


Fig. 13.3 Illustration of the integrated histogram method. The solid line represents the cumulative distribution function, $F(x)$, of the PDF $f(x)$, shown as a dotted line. The random number R is produced with a uniform PDF in the range [0, 1]. The method outputs x_{out}, which corresponds to the middle of bin m satisfying the condition (13.30). The functions $f(x)$ and $F(x)$ are here plotted with 1,000 bins to give the illusion of a continuous function.

If the histogram is obtained from a measurement, one can only hope that the statistical accuracy and the binning are both sufficient to obtain a reasonable representation of $f(x)$. To simplify, let us assume the histogram is partitioned in n bins of equal width Δx in the range $[x_{\min}, x_{\max}]$. Formally, the content of each bin, $h_x(i)$, with $i = 1,\ldots, n$, is given by the integral of the function $f(x)$ over the interval spanned by the bin:

$$h_x(i) = \int_{x_{\min,i}}^{x_{\max,i}} f(x)\,dx, \tag{13.28}$$

where $x_{\min,i} = x_{\min} + (i-1) \times \Delta x$ and $x_{\max,i} = x_{\min} + i \times \Delta x$. In practice, if the bins are sufficiently narrow, one may replace the preceding integral by $f(x_i)\Delta x$ where x_i is the midpoint of each bin. It may also be sufficient to use the (average) value of the function in the bin rather than the area under the curve. The definite integral $F(x) - F(x_{\min})$ may then be replaced by the sum

$$H_x(m) \equiv \sum_{i=1}^{m} h_x(i). \tag{13.29}$$

In the original inversion method, one uses a linear fraction $F(x) = R(F(x_{\max}) - F(x_{\min}))$ determined by a random number R to obtain a value x. This linear fraction is here replaced by $R \times H_x(n)$ or simply R if the sum $H_x(n)$ is normalized to unity. One must then find the value m satisfying

$$H_x(m-1) \leq R \times H_x(n) < H_x(m). \tag{13.30}$$

The generated value x may then be chosen has the middle of bin m. Alternatively, one can carry out a linear or quadratic interpolation based on nearby bins $H_x(m \pm 1)$. For sufficiently narrow bins, the method produces random deviates that mimic the function $f(x)$ (or the histogram representing the function).

The search for the value m that satisfies the preceding condition is a weakness of the method. If the number of bins, n, is large, the task of finding which value m satisfies the condition can be rather CPU intensive. A linear search, that is, proceeding by systematic inspection of all values of m starting from 1, takes an average of $n/2$ steps. However,

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

a binary tree search requires an average number of steps that scales as $\ln n$ and is thus considerably faster for histograms with a large number of bins.

13.3.6 Multidimensional Pseudorandom Generation

The simultaneous generation of random numbers for a system involving n variables may be handled as the random selection or production of an n-dimension vector $\vec{x}$ in an n-dimensional space. There are two basic methods, each with several variations, to handle the generation of vectors $\vec{x}$ in an n-dimensional space.

The first method assumes the generation of numbers can be factorized (at least approximately) whereas the second method is applicable whenever the range of the components exhibit a simple hierarchical relation.

Factorization Method

Production of random numbers distributed according to a function $f(\vec{x})$ with the **factorization method** is possible provided there exists a function $g(\vec{x})$ everywhere equal or larger than $f(\vec{x})$ and which can be factorized into a product of n distinct functions:

$$g(\vec{x}) = g_1(x_1) \times g_2(x_2) \times \cdots \times g_n(x_n) = \prod_{i=1}^{n} g_i(x_i), \tag{13.31}$$

where x_i, $i = 1, 2, n$ are the components of the n-dimensional vector $\vec{x}$. Each of the functions $g_i(x_i)$ may be simple or composite. The generation of vectors proceeds as follows:

1. First generate n x_i values independently using each of the g_i functions.
2. Generate a random number R in the range [0, 1] and compare it to the ratio $f(x)/g(x)$; if R is smaller than the ratio, retain the point, otherwise repeat step 1.

If the function $f(x)$ is itself factorable as a product of n functions (one for each dimension), it then suffices to set $g(x) = f(x)$ and the method reduces to step 1.

Hierarchical Boundary Conditions Method

The generation of random numbers can become an arbitrarily complicated problem if the ranges of the n components of the vector $\vec{x}$ are interdependent. In the simplest cases, the ranges of the n components are independent of each other and the generation of $\vec{x}$ then takes place in an n-dimensional hypercube. But there can be systems such that the generation of the different components of $\vec{x}$ have intricate interdependencies. This would be the case, for instance, if the range of x_3 depends on x_2 whose range itself depends on the value of x_1 or other values, etc.

The hierarchical boundary conditions method is useful if the boundaries of allowed regions can be written in a form where the range of x_1 is known, the allowed range of x_2 depends only on x_1, that of x_3 only on x_1 and x_2, and so on, until x_n whose range may depend on all the preceding variables. Under such conditions, it may be possible to find

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

a function $g(x)$ that can be integrated over x_2 through x_n to yield a simple function of x_1, according to which x_1 is selected. Having done that, x_2 is selected according to a distribution which now depends on x_1, but with x_3 through x_n integrated over. The procedure is continued until x_n is reached. In the end, the ratio $f(x)/g(x)$ is used to determine whether to retain the point.

13.3.7 Gaussian Generator

By virtue of the central limit theorem, a superposition of n random variables (i.e., a sum) leads to a Gaussian distribution in the limit $n \to \infty$. The Gaussian PDF is thus of special interest since it may be used to represent the fluctuations of a wide variety of processes and systems. One indeed finds that many random observables and systems can be described to a very good approximation with a Gaussian PDF. However, the generation of Gaussian deviates is not a trivial task. Indeed, first note that the Gaussian distribution cannot be integrated analytically. This implies one cannot use the inversion method and must resort to alternative techniques. Obviously, the rejection method is readily applicable for a Gaussian. However, it is quite inefficient and slow because the function has tails extending to infinity. Alternatively, one might consider using the central limit theorem itself since the sum of n numbers generated according to a uniform random generator in the range $[0, 1]$ has a Gaussian distribution in the large n limit. But this method is also rather inefficient and CPU-intensive given the production of a single Gaussian deviate as a sum requires the generation of a large number of uniform deviates. A much more efficient technique is based on the simple mathematical construction discussed in the text that follows.

A Gaussian PDF depends on two parameters: a centroid μ and a width σ. One might thus have the impression that an infinite number of different generators might be required to match arbitrary values of μ and σ. This is not the case, however, since any Gaussian can be mapped onto the standard normal distribution $\phi(z)$ defined such that the probability of finding z in the interval $[z, z + dz]$ is given by

$$\phi(z)dz = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) dz. \tag{13.32}$$

A simple change of variable $x = \sigma z + \mu$ thus provides for a Gaussian PDF with arbitrary mean μ and standard deviation σ. Indeed, since $z = (x - \mu)/\sigma$, and $dz = dx/\sigma$, one gets

$$p_G(x)dx = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx. \tag{13.33}$$

Gaussian deviates with arbitrary mean μ and standard deviation σ may thus be obtained by scaling random numbers z generated with a standard normal distribution according to $x = \sigma z + \mu$.

Normally distributed random numbers can be efficiently produced with an algorithm devised by Box and Muller [57]. Although a one-dimensional Gaussian PDF is not integrable, a simple transformation enables the integration of a two-dimensional Gaussian. It is thus possible to efficiently produce pairs of Gaussian deviates. Indeed, consider the

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

two-dimensional (standard) normal distribution defined as follows:

$$\phi_2(x, y) = \frac{1}{2\pi} \exp\left(-\frac{x^2 + y^2}{2}\right) = \phi(x)\phi(y). \tag{13.34}$$

The technique introduced by Box and Muller is based on the transformation

$$x = \sqrt{-2 \ln r} \cos 2\pi\phi, \tag{13.35}$$

$$y = \sqrt{-2 \ln r} \sin 2\pi\phi, \tag{13.36}$$

where r and ϕ are two uniformly distributed deviates in the semi-open range]0, 1]. The applicability of the method is readily verified since by definition of the variables x and y one gets

$$x^2 + y^2 = -2 \ln r, \tag{13.37}$$

$$\exp\left(-\frac{x^2 + y^2}{2}\right) = \exp(\ln r) = r. \tag{13.38}$$

Values of r and ϕ uniformly distributed in the range]0, 1] thus indeed map onto Gaussian deviates x and y. It is consequently possible to write a computer function that produces two Gaussian deviates per call to the function. The method is very CPU-efficient because it requires the generation of (only) two uniform deviates to produce two Gaussian deviates.

13.4 Selected Examples

We consider a few practical examples of random number generation commonly encountered in nuclear and particle physics.

13.4.1 Generation of Random Two-Dimensional Vectors

Let us begin by considering the generation of random vectors $\vec{r}$ in a two-dimensional configuration space. Obviously, two random numbers will be required for each random vector. However, one has a choice of representation: should one express the vectors in Cartesian coordinates or in polar coordinates? In the first case, one would have to generate two numbers r_x and r_y corresponding to positions along axes x and y. In the second case, one needs to produce a radius, r, and a polar angle θ.

The choice of representation shall depend on the specificities of the problem considered, as illustrated in Figure 13.4. For rectangular geometry involving, for instance, the simulation of particles entering detectors of rectangular cross section, it is obviously simplest to generate numbers x and y in Cartesian coordinates corresponding to the position of the particle's entry point along the face of the detector. On the other hand, if the vector $\vec{r}$ represents a displacement bound in some range $r_{\min} \leq |\vec{r}| < r_{\max}$, it might be more convenient to generate vectors in terms of radii r and polar angles θ.

To generate a random position uniformly in a rectangle defined by two corners $(x_{\min}, y_{\min})$ and $(x_{\max}, y_{\max})$, it is simplest to use rectangular coordinates as illustrated in

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

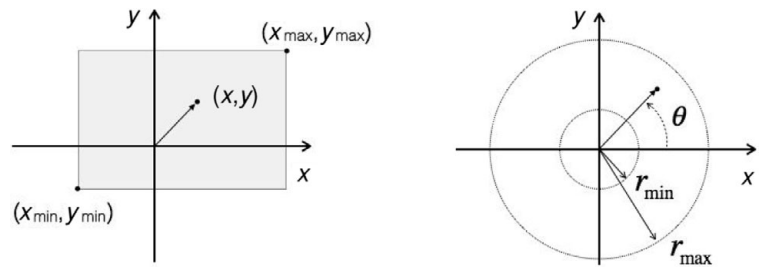


Fig. 13.4 Illustration of two geometries used for the generation of 2D random vectors. (a) Rectangular geometry, (b) circular geometry.

Figure 13.4a:

$$x = x_{\min} + r_1 \times (x_{\max} - x_{\min}), \tag{13.39}$$

$$y = y_{\min} + r_2 \times (y_{\max} - y_{\min}), \tag{13.40}$$

where r_1 and r_2 are random numbers uniformly distributed in the range [0, 1].

The generation of random displacements expressed in polar coordinates constitutes a more interesting case. If all directions are uniformly probable, the direction may be randomly produced with $\theta = 2\pi r_1$ in which r_1 is a uniform deviate in the range [0, 1]. For displacement magnitudes uniformly distributed in the range $r \leq r_{\max}$, one might initially be tempted to use a linear expression such as $r_2 \times r_{\max}$. Note, however, that in polar coordinates, the element of area is $rd\theta\, dr$. Consequently, uniform sampling in r would lead to a decreasing number of "hits" per unit of area for increasing values of radius r. However, given the element of area can also be expressed as $d\theta d\left(r^2\right)$, the sampling should be uniform in r^2 rather than r and one writes

$$r^2 = r_{\min}^2 + r_2 \times \left(r_{\max}^2 - r_{\min}^2\right), \tag{13.41}$$

where r_2 is a uniform deviate in the range [0, 1]. Formally, this can be derived as follows: since by assumption the displacement density is set to be uniform per unit of area, the displacement density can be written

$$\frac{dN}{dxdy} = \frac{1}{A_{\text{TOT}}}, \tag{13.42}$$

where A_{TOT} is a constant denoting the total area under consideration, that is, the fiducial area in which displacements can be generated. Transforming the (x, y) coordinates in polar coordinates (r, θ), one gets

$$\frac{dN}{drd\theta} = \frac{dN}{dxdy}\left|\frac{\partial(x, y)}{\partial(r, \theta)}\right| = \frac{r}{A_{\text{TOT}}}. \tag{13.43}$$

The PDF determining the r distribution is thus

$$f(r) = \frac{dN}{dr} = \frac{2\pi r}{A_{\text{TOT}}}. \tag{13.44}$$

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

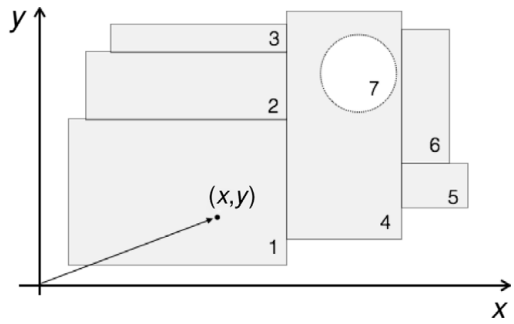


Fig. 13.5 Illustration of random number generation with complex masks featuring a combination of inclusion (gray) and exclusion regions (white).

Assuming minimum and maximum radii $r_{\min}$ and $r_{\max}$, let us proceed to derive a formula for the generation of r values using the inversion method. Integration of $f(r)$ yields $F(r)$ as follows:

$$\int_{r_{\min}}^{r} f(r)dr = F(r) - F(r_{\min}) = \frac{\pi}{A_{\mathrm{TOT}}}\left(r^2 - r_{\min}^2\right). \tag{13.45}$$

Finally, inversion of $F(r)$ produces the anticipated result:

$$r = \sqrt{r_2\left(r_{\max}^2 - r_{\min}^2\right) + r_{\min}^2}, \tag{13.46}$$

where r_2 is a random uniform deviate in the range [0, 1]. The displacement vector is thus $\overrightarrow{\Delta r} = (r\cos\theta, r\sin\theta)$.

We have assumed that the radial displacements would generate a uniform density per unit of area. This is unlikely to be the case in general. A more appropriate probability density $f(r)$ that matches the specificities of a given system has to be identified and the size r of the displacement calculated accordingly. For instance, if $f(r)$ is constant in the range $[r_{\min}, r_{\max}]$, than radii may be generated according to $r = r_2\,(r_{\max} - r_{\min}) + r_{\min}$. If instead, $f(r) = 1/r$, then one gets $r = \exp\left[r_2\,(\ln r_{\max} - \ln r_{\min}) + \ln r_{\min}\right]$.

Perfectly rectangular or circular geometries are obviously uncommon in real-world situations. And it is in fact where Monte Carlo methods find their most interesting and powerful uses. In two-dimensional geometries involving nonrectangular boundaries (i.e., curved boundaries) or forbidden regions, one must exercise judgment and creativity as to what shall constitute the most CPU-effective generation process. As illustrated in Figure 13.5, one might, for instance, use a large rectangle that encloses the fiducial region (i.e., the region of interest) and use "masks" to dismiss points produced in forbidden regions.

In the case illustrated in Figure 13.5, this amounts to having mathematical formulation of the boundaries, that is, minimum and maximum value expressions for both the x and y. This can be accomplished with one or multiple functions. Alternatively, one can partition the sample space and use a two-step approach: for instance, if all positions are uniformly probable (in the allowed regions), one first randomly selects in which partition the position should be generated according to the relative size of the area of the various partitions, and subsequently generate a specific position within the selected partition.

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

13.4.2 Generation of Random Three-Dimensional Vectors

As with two-dimensional vectors, one must choose a coordinate system for the generation of random three-dimensional vectors. The choice of Cartesian coordinates, cylindrical coordinates, spherical coordinates, or other coordinate systems is determined by convenience and the symmetry of the problem at hand. Since the generation in Cartesian and cylindrical coordinates is rather similar to the generation of two-dimensional vectors, we focus on the generation of vectors using spherical coordinates, which requires the production of a radius r, a polar angle θ, and an azimuthal angle ϕ.

Let us consider all positions within a sphere of a radius $r_{\max}$ to be equally probable. The radius r is to be generated in the range $[0, r_{\max}]$, while the polar and azimuthal angles are in the ranges $0 \leq \theta \leq \pi$ and $0 \leq \phi < 2\pi$, respectively. Since all positions are equally probable, the number of points per unit of volume $r^2 \sin\theta d\theta d\phi dr$ must be generated uniformly. Rewriting the element of volume $d\phi d(\cos\theta)d(r^3)/3$, we conclude that equally probable positions, within a sphere of radius $r_{\max}$, may be achieved by generating values for ϕ, $\cos\theta$, and r^3 using three random numbers r_i, $i = 1, 2, 3$ in the range $[0, 1]$ as follows:

$$\phi = 2\pi r_1, \tag{13.47}$$

$$\cos\theta = 2r_2 - 1, \tag{13.48}$$

$$r^3 = r_3 r_{\max}^3. \tag{13.49}$$

Note that by generating $\cos\theta$ in the range $[-1, 1]$, one ensures that all polar angles in the range $[0, \pi]$ are produced, and that all elements of volumes, being proportional to $d\cos\theta$, are equally probable. Likewise, the generation of r^3 with a uniform PDF, rather than r, insures that the point density remains constant throughout the volume of the sphere.

The generation of three-dimensional vectors whose probability are functions of the coordinates r, θ, and ϕ should be treated according to the specificities of the system to be simulated. Section 14.1.1 discusses the generation of momentum vectors according to a Maxwell–Boltzmann distribution as an example of nontrivial dependence on the length (radius) of the vector. Dependencies on θ and ϕ are obviously also possible. For instance, anisotropic flow production in heavy ion collisions may be simulated with a simple function of the azimuthal angle ϕ presented in §14.1.5.

13.4.3 Weighing Method

The generation of one or several random variables according to a specific PDF is not always practical or even desirable because the PDF might be too complicated or because the geometry and boundary conditions at hand are too intricate. It remains, nonetheless, that the relative frequency of random variables may have to obey specific probability density in order to represent the particular phenomenon being modeled. The **weighing method** may then provide a simple solution.

Let us consider the generation of "events" consisting of n random numbers $\vec{x} = (x_1, x_2, \ldots, x_n)$ in some domain D. Rather than generating each $\vec{x}$ according to a complicated procedure, one may elect to produce each of the values x_i, $i = 1, \ldots, n$ according

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

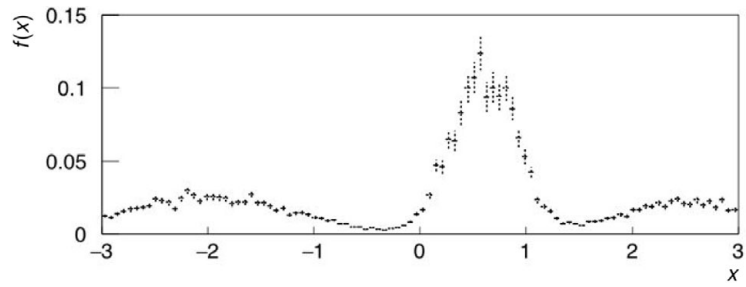


Fig. 13.6 Random triple Gaussian deviates generated with the weighing method according to Eq. (13.50).

to uniform generators, each in some appropriate or convenient range. The technique then involves the assignment of a weight to each $\vec{x}$ produced according to some weight function $W(\vec{x})$. The variable or variables required are first generated according to uniform distributions in their respective ranges of interest. Once the variables are generated, one verifies whether the event lies within the domain of interest. If it does not, the event is simply ignored and the procedure repeated. If the event is deemed of interest, one calculates its weight according $W(\vec{x})$. Then, rather than counting each event as a unit, one increments histograms of interest in proportion to the weight $W(\vec{x})$ of the event.

Consider, as a simple example, the production of events consisting of a single random value x to be distributed according to a triple Gaussian distribution in the range $-3 < x \leq 3$ for specific values of the means μ_i, widths σ_i, and normalizations f_i for $i = 1, \ldots, 3$. Let us ignore the fact that there exists several techniques to obtain Gaussian deviates and generate values x according to a uniform distribution in the range $-3 < x \leq 3$ and calculate for each value of x, the weight W according to

$$W(x) = \sum_{i=1}^{3} f_i \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right]. \tag{13.50}$$

Figure 13.6 displays a triple Gaussian distribution obtained with 10,000 events and $\mu_1 = -2$, $\sigma_1 = 0.75$; $\mu_2 = 0.6$, $\sigma_2 = 0.30$; $\mu_3 = 2.5$, $\sigma_3 = 0.60$.

The weight technique is particularly useful for the simulation of events involving several variables that are correlated but difficult to generate ab initio. It also presents the advantage that given a set of existing events, one can apply weights a posteriori and study, for instance, the effect of different weight functions on measured distributions.

Exercises

13.1 Use the inversion method to create a generator producing random numbers according to a parabolic PDF, $f(x) = ax^2$. First find the inversion expression and then implement it in a computer program. Use your program to generate a histogram of the PDF

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press

in the range $-10 \leq x < 10$, and inspect the histogram for 100, 500, 1,000, 10,000 or more events.

13.2 Use the acceptance/rejection method to create a generator producing numbers according to $f(x) \propto x \exp(-x^2/2\sigma^2)$ in the range [0, 10].

13.3 Use the acceptance/rejection method to create a generator producing numbers according to $f(x, y) \propto x^2 y^2 \exp\left(-\left(x^2 + y^2\right)/2\sigma^2\right)$ in the domain $-4 \leq x < 4$, $-5 \leq y < 5$, for $\sigma = 1.5$.

13.4 Use the MC method defined by Eq. (13.7) to calculate the area of a circle and determine the value of π.

13.5 Write a function that produces Gaussian deviates based on the Box and Muller method.

13.6 Write a function that produces Gaussian deviates based on the central limit theorem and uniform deviates in the range [0, 1]. Study the precision of the method by varying the number of deviates used in the sum.

13.7 Write a function that produces Gaussian random numbers based on the integrated histogram inversion method. Hint: Select a domain of applicability and produce a table or histogram containing a running integral of the Gaussian distribution.

13.8 Find a method to generate Gaussian random numbers in the x–y plane with expectation values $E[x] = \mu_x$, $E[y] = \mu_y$, and variances $\mathrm{Var}[x] = \pi y$, and $\mathrm{Var}[y] = \pi x$.

13.9 Using the generator written in Problem 13.1, study the precision of the mean, variance, skewness of the distributions obtained with 100, 500, 1,000, 10,000 or more events.

https://doi.org/10.1017/9781108241922.017 Published online by Cambridge University Press