11

Monte Carlo techniques

So far we have been focussing on how particle codes work once the particles are launched. We've talked about how they are moved, and how self-consistent forces on them are calculated. What we have not addressed is how they are launched in an appropriate way in the first place, and how particles are reinjected into a simulation. We've also not explained how one decides statistically whether a collision has taken place to any particle and how one would then decide what scattering angle the collision corresponds to. All of this must be determined in computational physics and engineering by the use of random numbers and statistical distributions.[1] Techniques based on random numbers are called by the name of the famous casino at Monte Carlo.

11.1 Probability and statistics

11.1.1 Probability and probability distribution

Probability, in the mathematically precise sense, is an idealization of the repetition of a measurement, or a sample, or some other test. The result in each individual case is supposed to be unpredictable to some extent, but the repeated tests show some average trends that it is the job of probability to represent. So, for example, the single toss of a coin gives an unpredictable result: heads or tails; but the repeated toss of a (fair) coin gives on average equal numbers of heads and tails. Probability theory describes that regularity by saying the probability of heads and tails is equal. Generally, the probability of a particular class of outcomes (e.g. heads) is defined as the *fraction of the outcomes*, in a very

[1] S. Brandt (2014), *Data Analysis Statistical and Computational Methods for Scientists and Engineers*, fourth edition, Springer, New York, gives a much more expansive introduction to statistics and Monte Carlo techniques.

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

large number of tests, that are in the particular class. For a fair coin toss, the probability of heads is the fraction of outcomes of a large number of tosses that is heads, 0.5. For a six-sided die, the probability of getting any particular value, say 1, is the fraction of rolls that come up 1, in a very large number of tests. That will be one-sixth for a fair die. In all cases, because probability is defined as a *fraction*, the sum of probabilities of all possible outcomes must be unity.

More usually, in describing physical systems we deal with a *continuous* real-valued outcome, such as the speed of a randomly chosen particle. In that case the probability is described by a "probability distribution" $p(v)$, which is a function of the random variable (in this case velocity v). The probability of finding that the velocity lies in the range $v \rightarrow v + dv$ for small dv is then equal to $p(v)dv$. In order for the sum of all possible probabilities to be unity, we require

$$\int p(v)dv = 1. \tag{11.1}$$

Each individual sample[2] might give rise to more than one value. For example the velocity of a sampled particle might be a three-dimensional vector $\boldsymbol{v} = (v_x, v_y, v_z)$. In that case, the probability distribution is a function in a multidimensional parameter-space, and the probability of obtaining a sample that happens to be in a multidimensional element d^3v at $\boldsymbol{v}$ is $p(\boldsymbol{v})d^3v$. The corresponding normalization is

$$\int p(\boldsymbol{v})d^3v = 1. \tag{11.2}$$

Obviously, what this shows is that if our sample consists of randomly selecting particles from a velocity distribution function $f(\boldsymbol{v})$, then the corresponding probability function is simply

$$p(\boldsymbol{v}) = f(\boldsymbol{v}) / \int f(\boldsymbol{v})d^3v = f(\boldsymbol{v})/n, \tag{11.3}$$

where n is the particle density. So the normalized distribution function is the velocity probability distribution.

The *cumulative* probability function can be considered to represent the probability that a sample value is less than a particular value. So for a single-parameter distribution $p(v)$, the cumulative probability is

$$P(v) = \int_{-\infty}^{v} p(v')dv'. \tag{11.4}$$

[2] Statisticians use the generic word "sample" to refer to the particular result of a single test or measurement.

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

In multiple dimensions, the cumulative probability is a multidimensional function that is the integral in all the dimensions of the probability distribution:

$$P(\boldsymbol{v}) = P(v_x, v_y, v_z) = \int_{-\infty}^{v_x} \int_{-\infty}^{v_y} \int_{-\infty}^{v_z} p(\boldsymbol{v}') d^3 v'. \tag{11.5}$$

Correspondingly, the probability distribution is the derivative of the cumulative probability: $p(v) = dP/dv$, or $p(\boldsymbol{v}) = \partial^3 P/\partial v_x \partial v_y \partial v_z$.

11.1.2 Mean, variance, standard deviation, and standard error

If we make a large number N of individual measurements of a random value from a probability distribution $p(v)$, each of which gives a value v_i, $i = 1, 2, \ldots, N$, then the *sample mean* value of the combined sample N is defined as

$$\mu_N = \frac{1}{N} \sum_{i=1}^{N} v_i. \tag{11.6}$$

The *sample variance* is defined[3] as

$$S_N^2 = \frac{1}{N-1} \sum_{i=1}^{N} (v_i - \mu_N)^2. \tag{11.7}$$

The *sample standard deviation* is S_N, the square root of the variance, and the *sample standard error* is $S_N/\sqrt{N}$. The mean is obviously a measure of the average value, and the variance or standard deviation is a measure of how spread out the random values are. They are the simplest unbiassed estimates of the moments of the distribution. These moments are properties of the *probability distribution* not of the particular sample. The *distribution mean*[4] is defined as

$$\mu = \int v p(v) dv \tag{11.8}$$

and the *distribution variance* is

$$S^2 = \int (v - \mu)^2 p(v) dv. \tag{11.9}$$

[3] Division by the factor $N - 1$ rather than N makes this formula an unbiassed estimate of the distribution variance. One way to understand this is to recognize that the number of degrees of freedom of $\sum_1^N (v_i - \mu_N)^2$ is $N - 1$, not N. Using $N - 1$ is sometimes called "Bessel's correction".

[4] Often called the "expectation" of v.

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

Obviously for large N we expect the sample mean to be approximately equal to the distribution mean and the sample variance equal to the distribution variance.

A finite-size sample will not have a mean exactly equal to the distribution mean because of statistical fluctuations. If we regard the sample mean μ_N as itself being a random variable, which changes from one total sample of N tests to the next total sample of N tests, then it can be shown[5] that the probability distribution of μ_N is approximately a Gaussian with standard deviation equal to the standard error $S_N/\sqrt{N}$. That is one reason why the Gaussian distribution is sometimes called the "normal" distribution. The Gaussian probability distribution in one dimension has only two[6] independent parameters μ and S.

11.2 Computational random selection

Computers can generate *pseudo*-random numbers, usually by doing complicated non-linear arithmetic starting from a particular "seed," number (or strictly a seed "state," which might be multiple numbers). Each successive number produced is actually completely determined by the algorithm, but the sequence has the appearance of randomness, in that the values v jump around in the range $0 \le v \le 1$, with no apparent systematic trend to them. If the random-number generator is a good generator, then successive values will not have statistically detectable dependence on the prior values, and the

[5] **Enrichment: Central Limit Theorem**. It is not straightforward to prove that the distribution becomes Gaussian. But it is fairly easy to show that the variance of the sample mean is the variance of the distribution divided by N. From the definition of μ_N one can immediately deduce that

$$(\mu_N - \mu)^2 = \left(\frac{1}{N}\sum_1^N (v_i - \mu)\right)^2 = \frac{1}{N^2}\sum_{i,j}^N (v_i - \mu)(v_j - \mu).$$

Take the expectation $\langle \dots \rangle$ of this quantity to obtain the variance of the distribution of sample means:

$$\langle (\mu_N - \mu)^2 \rangle = \frac{1}{N^2}\sum_{i,j}^N \langle (v_i - \mu)(v_j - \mu)\rangle = \frac{1}{N^2}\sum_i^N \langle (v_i - \mu)^2\rangle = \frac{S^2}{N}.$$

The first equality, taking the expectation inside the sum, is a simple property of taking the expectation: the expectation of a sum is the sum of the expectations. The second equality uses the fact that $\langle (v_i - \mu)(v_j - \mu)\rangle = 0$ for $i \neq j$ because the quantities $(v_i - \mu)$ are statistically independent and have zero mean. That is sufficient to yield the required result. Our estimate for the distribution variance is $S^2 = S_N^2$. So the unbiassed estimate for the variance of μ_N is $\langle (\mu_N - \mu)^2\rangle = S_N^2/N$. The standard error is the square root of this quantity.

[6] An un-normalized Gaussian distribution has three, including the height.

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

distribution of values in the range will be uniform, representing a probability distribution $p(v) = 1$. Many languages and mathematical systems have library functions that return a random-number. Not all such functions are "good" random number generators. (The built-in C functions are notoriously not good.) One should be wary for production work. It is also extremely useful, for example for program debugging, to be able to repeat a pseudo-random calculation, knowing the sequence of "random" numbers you get each time will be the same. What you must be careful about, though, is that if you want to improve the accuracy of a calculation by increasing the number of samples, it is essential that the samples be independent. Obviously, that means the random numbers you use must *not* be the same ones you already used. In other words, the seed must be different. This goes also for parallel computations. Different parallel processors should normally use different seeds.

Now obviously if our computational task calls for a random number from a uniform distribution between 0 and 1, $p(v) = 1$, then using one of the internal or external library functions is the way to go. However, usually we will be in a situation where the probability distribution we want to draw from is *non-uniform*, for example a Gaussian distribution, an exponential distribution, or some other function of value. How do we do that?

We use two related random variables; call them u and v. Variable u is going to be uniformly distributed between 0 and 1. (It is called a "uniform deviate".) Variable v is going to be related to u through some one-to-one functional relation. Now if we take a particular sample value drawn from the uniform deviate, u, there is a corresponding value v. What's more, we know that the fraction of drawn values that are in a particular u-element du is equal to the fraction of values that are in the corresponding v-element dv. Consequently, recognizing that those fractions are $p_u(u)du$ and $p_v(v)dv$, respectively, where p_u and p_v are the respective probability distributions of u and v, we have

$$p_u(u)du = p_v(v)dv \qquad \Rightarrow \quad p_v(v) = p_u(u)\left|\frac{du}{dv}\right| = \left|\frac{du}{dv}\right|. \tag{11.10}$$

The final equality uses the fact that $p_u = 1$ for a uniform deviate.

Therefore, if we are required to find random values v from a probability distribution p_v, we simply have to find a functional relationship between v and u that satisfies $p_v(v) = |du/dv|$. But we know of a function already that provides this property. Consider the cumulative probability $P_v(v) = \int^v p_v(v')dv'$. It is monotonic, and ranges between 0 and 1. Therefore we may choose to write

$$u = P_v(v) \qquad \text{for which} \quad \frac{du}{dv} = p_v(v). \tag{11.11}$$

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

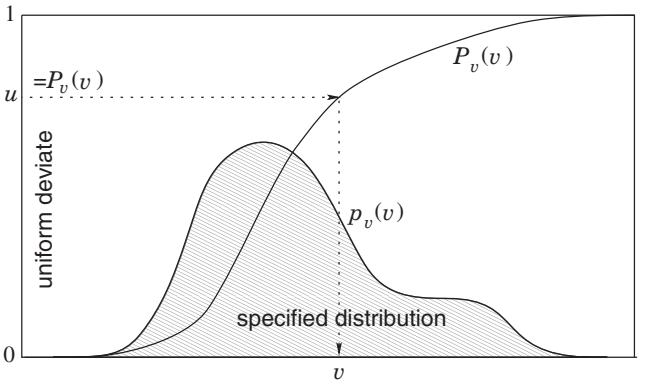

Figure 11.1 To obtain numerically a random variable v with specified probability distribution p_v (not to scale), calculate a table of the function $P_v(v)$ by integration. Draw a random number from uniform deviate u. Find the v for which $P_v(v) = u$ by interpolation. That's the random v.

So if $u = P_v(v)$, the v variable will be randomly distributed with probability distribution $p_v(v)$. We are done. Actually not quite done, because the process of choosing u and then finding the value of v which corresponds to it requires us to invert the function $P_v(v)$. That is

$$v = P_v^{-1}(u). \tag{11.12}$$

Figure 11.1 illustrates this process. It is not always possible to invert the function analytically, but it is always possible to do it numerically. One way is by root finding, e.g. bisection. Since P_v is monotonic, for any u between 0 and 1, there is a single root v of the equation $P_v(v) - u = 0$. Provided that we can find that root quickly, then given u we can find v. One way to make the root-finding quick is to generate a table of values of v and $u = P_v(v)$, of length N_t, equally spaced *in* u (not in v). Then, given any u, the index of the point just below u is the integer value $i = u^*N_t$, and we can interpolate between it and the next point using the fractional value of u^*N_t.[7]

Rejection method Another way of obtaining random values from some specified probability distribution is by the "rejection method", illustrated in Fig. 11.2. This involves using a second random number to decide whether or not to retain the first one chosen. The second random number is used to weight

[7] Linear interpolation is then equivalent to representing p_v as a histogram. So adequate resolution may require a fairly large number N_t.

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

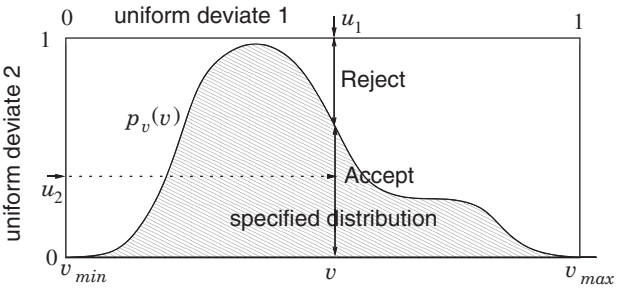

Figure 11.2 The rejection method chooses a v value randomly from a simple distribution (e.g. a constant) whose integral is invertible. Then a second random number decides whether it will be rejected or accepted. The fraction accepted at v is equal to the ratio of $p_v(v)$ to the simple invertible distribution. $p_v(v)$ must be scaled by a constant factor to be everywhere less than the simple distribution (1 here).

the probability of the first one. In effect this means picking points below the first scaled distribution, in the illustrated case of a rectangular distribution, uniformly distributed within the rectangle, and accepting only those that are below $p_v(v)$ (suitably scaled to be everywhere less than 1). Therefore some inefficiency is inevitable. If the area under $p_v(v)$ is, say, half the total, then twice as many total choices are needed, and each requires two random numbers, giving four times as many random numbers per accepted point. Improvement on the second inefficiency can be obtained by using a simply invertible function that fits $p_v(v)$ more closely. Even so, this will be slower than the tabulated function method, unless the random-number generator has very small cost.

Monte Carlo integration Notice, by the way, that this second technique shows exactly how "Monte Carlo integration" can be done. Select points at random over a line, or a rectangular area in two dimensions, or cuboid volume in three dimensions. Decide whether each point is within the area/volume of interest. If so, add the value of the function to be integrated to the running total, if not, not. Repeat. At the end multiply the total by the area/volume of the rectangle/cuboid divided by the number of random points examined (total, not just those that are within the area/volume). That's the integral. Such a technique can be quite an efficient way, and certainly an easy-to-program way, to integrate over a volume for which it is simple to decide whether you are inside it but hard to define systematically where the boundaries are. An example might be the volume inside a cube but outside a sphere placed off-center inside the cube. The method's drawback is that its accuracy increases

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

only like the inverse *square root* of the number of points sampled. So, if high accuracy is required, other methods may be much more efficient.[8]

11.3 Flux integration and injection choice

Suppose we are simulating a subvolume that is embedded in a larger region. Particles move in and out of the subvolume. Something interesting is being modeled within the subvolume, for example the interaction of some object with the particles. If the volume is big enough, the region outside the subvolume is relatively unaffected by the physics in the subvolume, then we know or can specify what the distribution function of particles is in the outer region, at the volume's boundaries. Assume that periodic boundary conditions are not appropriate, because, for example, they don't well represent an isolated interaction. How do we determine statistically what particles to inject into the subvolume across its boundary?

Suppose the volume is a cuboid shown in Fig. 11.3. It has six faces, each of which is normal to one of the coordinate axes, and located at $\pm L_x$, $\pm L_y$ or $\pm L_z$. We'll consider the face perpendicular to x, which is at $-L_x$, so that positive velocity v_x corresponds to moving *into* the simulation subvolume. We calculate the rate at which particles are crossing the face into the subvolume. If the distribution function is $f(\boldsymbol{v},\boldsymbol{x})$, then the flux density in the $+v_x$ direction is

$$\Gamma_x(\boldsymbol{x}) = \int\int\int_{v_x=0}^{\infty} v_x f(\boldsymbol{v},\boldsymbol{x}) dv_x dv_y dv_z \tag{11.13}$$

and the number entering across the face per unit time (the flux) is

$$F_{-L_x} = \int_{-L_y}^{L_y}\int_{-L_y}^{L_y} \Gamma_x(-L_x, y, z)\, dydz. \tag{11.14}$$

[8] **Enrichment: Quiet start and quasi-random selection.** When starting a particle-in-cell (PIC) simulation, the initial positions of the particles might be chosen using random numbers to decide their location. However, they then will have density fluctuations of various wavelengths that in plasmas may be *bigger* than are present after running the simulation for many steps. The reason for this discrepancy is that the feedback effect of the self-consistent electric potential tends to smooth out density fluctuations, so that in the fully developed simulation the noise level is lower than purely random. A simple way of saying the same thing is that individual particles *repel* others of the same type, preventing clumping of the particles. It is therefore often physically reasonable to start a PIC simulation with positions that are chosen to be more evenly spaced than purely random. Indeed, for some calculations it is advantageous (but non-physical) to start with density fluctuations even *lower* than the final level that would be present for a steady plasma. In either case, what is called a "quiet start" can be obtained by using what are called "quasi-random" numbers (see, e.g., *Numerical Recipes*) instead of (pseudo) random numbers. Quasi-random numbers are somewhat random, but much smoother in their distribution because each new number takes account of the already used numbers and tries to avoid being close to them. Successive numbers are thus correlated rather than uncorrelated. For Monte Carlo integration, such smoother distributions in space are also often highly appropriate and can give lower-noise results for the same number of samples, beating the weak, $1/\sqrt{N}$, decrease of fractional error.

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

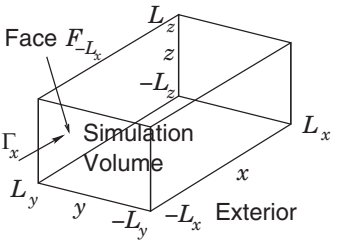

Figure 11.3 Simulating over a volume that is embedded in a wider external region, we need to be able to decide how to inject particles from the exterior into the simulation volume so as to represent statistically the exterior distribution.

Assume that the outer region is steady, independent of time. We proceed by evaluating the six fluxes F_j, using the above expressions and their equivalents for the six different faces. At each timestep of the simulation, we decide how many particles to inject into the subvolume from each face. The average number of particles to inject in a timestep Δt at face j is equal to $F_j \Delta t$. If this number is large,[9] then it may be appropriate to inject just that number (although dealing with the non-integer fractional part of the number needs consideration). But if it is of order unity or even smaller, then that does not correctly represent the statistics. In that case we need to decide statistically at each step how many particles are injected: 0, 1, 2 ....

It is a standard result of probability theory[10] that if events (in this case injections) take place randomly and uncorrelated with each other at a fixed

[9] Or if we wish to do "quiet injection" that is smoother than purely random.

[10] **Enrichment: The discrete Poisson distribution.** Suppose there is a very large number N of similar, uncorrelated, events (for example radioactive decays of atoms) waiting to occur. In a time duration far shorter than the waiting time (e.g. the half-life) of an individual event the probability that any one of them occurs is small, say $p = r/N$. Then r is the average total number of events occuring in this time duration. The total number of events actually occuring in a particular time sample is then an integer, and we want to find the probability of each possible integer. Any sample consists of N choices about the individual events: yes or no. The number of yes events is distributed as a binomial distribution, in which the probability of n yes events is given by the number of different ways to choose n out of N, which is $\frac{N!}{n!(N-n)!}$, times the probability of each specific arrangement of n yes events and $N-n$ no events, $p^n(1-p)^{N-n}$. The total is

$$p_n = \frac{N!}{n!(N-n)!} p^n (1-p)^{N-n} = \frac{r^n}{n!} \left(1 - \frac{r}{N}\right)^N \left[\frac{1}{N^n (1 - r/N)^n} \frac{N!}{(N-n)!}\right].$$

Now recognize that the limit for large N (but constant n) of the square bracket term is 1; while the limit of the term $\left(1 - \frac{r}{N}\right)^N$ is $\exp(-r)$. Therefore the probability of obtaining n total events, of a type that are completely uncorrelated ($N \to \infty$), when their average rate of occurrence is r, is

$$p_n = \frac{r^n}{n!} \exp(-r).$$

This is the discrete Poisson distribution.

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

average rate r (per sample, in this case per timestep) then the number n that happens in any particular sample is an integer random variable with "Poisson distribution": a discrete probability distribution

$$p_n = \exp(-r)r^n/n! \,. \tag{11.15}$$

The parameter giving the rate, r, is a real number, but the number for each sample, n, is an integer. One can rapidly verify that, since $\sum_{n=0}^{\infty} r^n/n! = \exp(r)$, the probabilities are properly normalized: $\sum_n p_n = 1$. The mean rate is $\sum_n np_n = r$ (as advertized). The variance, it turns out, is also r. So the standard deviation is $\sqrt{r}$. The value p_n gives us precisely the probability that n particles will need to be injected when the flux is $r = F_j$. So the first step in deciding injections is to select randomly a number to be injected, from the Poisson distribution eq. (11.15). There are various ways to do this (including library routines). The root-finding approach is easily applied, because the cumulative probability function $P_u(u)$ can be considered to consist of steps of height p_n at the integers n (and constant in between).

Next, we need to decide where on the surface each injection is going to take place. If the flux density is uniform, then we just pick randomly a position corresponding to $-L_y \le y \le L_y$ and $-L_z \le z \le L_z$. Non-uniform flux density, however, introduces another distribution function inversion headache. It's more work, but straightforward.

Finally, we need to select the actual velocity of the particle. Very importantly, the probability distribution of this selection is *not* just the velocity distribution function, or the velocity distribution function restricted to positive v_x. No, it is the *flux* distribution $v_x f(\boldsymbol{v}, \boldsymbol{x})$ weighted by the normal velocity v_x (for an x-surface) if positive (otherwise zero). If the distribution is separable, $f(\boldsymbol{v}) = f_x(v_x)f_y(v_y)f_z(v_z)$, as it is if it is Maxwellian, then the tangential velocities v_y, v_z, can be treated separately: select two different velocities from the respective distributions. And select v_x (positive) from a probability distribution proportional to $v_x f_x(v_x)$.

If f is not separable, then a more elaborate random selection is required. Suppose we have the cumulative probability distribution of velocities $P(v_x, v_y, v_z)$ for all interesting values of velocity. Notice that if v_{xmax}, v_{ymax}, v_{zmax} denote the largest relevant values of v_x, v_y and v_z, beyond which $f = 0$, then $P(v_x, v_{ymax}, v_{zmax})$ is a function of just one variable v_x, and is the cumulative probability distribution integrated over the entire relevant range of the other velocity components. That is, it is the one-dimensional cumulative probability distribution of v_x. We can convert it into a one-dimensional flux cumulative probability by performing the following integral (numerically, discretely):

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

$$F(v_x) = \int_0^{v_x} v'_x \frac{\partial}{\partial v'_x} P(v'_x, v_{ymax}, v_{zmax})\, dv'_x \qquad (11.16)$$
$$= v_x P(v_x, v_{ymax}, v_{zmax}) - \int_0^{v_x} P(v'_x, v_{ymax}, v_{zmax})\, dv'_x.$$

Afterwards, we can normalize $F(v_x)$ by dividing by $F(v_{xmax})$, ariving at the cumulative flux-weighted probability for v_x.

We then proceed as follows.

1. Choose a random v_x from its cumulative flux-weighted probability $F(v_x)$.
2. Choose a random v_y from its cumulative probability for the already chosen v_x, namely $P(v_x, v_y, v_{zmax})$ regarded as a function only of v_y.
3. Choose a random v_z from its cumulative probability for the already chosen v_x and v_y, namely $P(v_x, v_y, v_z)$ regarded as a function only of v_z.

Naturally for other faces y and z one has to start with the corresponding velocity component and cycle round the indices. For steady external conditions all the cumulative velocity probabilities need to be calculated only once, and then stored for subsequent timesteps.

Discrete-particle representation An alternative to this continuum approach is to suppose that the external distribution function is represented by a perhaps large number, N, (maybe millions) of representative "particles" distributed in accordance with the external velocity distribution function. Particle k has velocity $\boldsymbol{v}_k$ and the density of particles in phase-space is proportional to the distribution function, that is to the probability distribution. Then if we wished to select randomly a velocity from the particle distribution we simply arrange the particles in order and pick one of them at random. However, when we want the particles to be flux-weighted, in normal direction $\hat{\boldsymbol{n}}$ say, we must select them with probability proportional to $v_n = \hat{\boldsymbol{n}}.\boldsymbol{v}$ (when positive, and zero when negative). Therefore, for this normal direction we must consider each particle to have appropriate weight. We associate each particle with a real[11] index r so that when $r_k \le r < r_{k+1}$ the particle k is indicated. The interval length allocated to particle k is chosen proportional to its weight, so that $r_{k+1} - r_k \propto \hat{\boldsymbol{n}}.\boldsymbol{v}_k$. Then the selection consists of a random-number draw x, multiplied by the total real index range and indexed to the particle and hence to its velocity: $x(r_{N+1} - r_1) + r_1 = r \to k \to \boldsymbol{v}_k$. The discreteness of the particle distribution will generally not be an important limitation for a process that already relies on random particle representation. The position of injection

[11] Or double-precision if N is very large.

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

will anyway be selected differently even if a representative particle velocity is selected more than once.

Worked example. High-dimensionality integration

Monte Carlo techniques are commonly used for high-dimensional problems; integration is perhaps the simplest example. The reasoning is approximately this. When there are d dimensions, the total number of points in a grid whose size is M in each coordinate direction is $N = M^d$. The fractional uncertainty in estimating a one-dimensional integral of a function with only isolated discontinuities, based upon evaluating it at M grid points, may be estimated as $\propto 1/M$. If this estimate still applies to multiple-dimension integration (and this is the dubious part), then the fractional uncertainty is $\propto 1/M = N^{-1/d}$. By comparison, the uncertainty in a Monte Carlo estimate of the integral based on N evaluations is $\propto N^{-1/2}$. When d is larger than 2, the Monte Carlo square-root convergence scaling is better than the grid estimate. And if d is very large, Monte Carlo is much better. Is this argument correct? Test it by obtaining the volume of a four-dimensional hypersphere by numerical integration, comparing a grid technique with Monte Carlo.

A four-dimensional sphere of radius 1 consists of all those points for which $r^2 = x_1^2 + x_2^2 + x_3^2 + x_4^2 \leq 1$. Its volume is known analytically; it is $\pi^2/2$. Let us evaluate the volume numerically by examing the unit hypercube $0 \leq x_i \leq 1$, $i = 1, \ldots, 4$. It is $1/2^4 = 1/16$th of the hypercube $-1 \leq x_i \leq 1$, inside of which the hypersphere fits; so the volume of the hypersphere that lies within the $0 \leq x_i \leq 1$ unit hypercube is $1/16$th of its total volume; it is $\pi^2/32$. We calculate this volume numerically by discrete integration as follows.

A deterministic (non-random) integration of the volume consists of constructing an equally spaced lattice of points at the center of cells that fill the unit cube. If there are M points per edge, then the lattice positions in the dimension i ($i = 1, \ldots, 4$) of the cell-centers are $x_{i,k_i} = (k_i - 0.5)/M$, where $k_i = 1, \ldots, M$ is the (dimension-i) position index. We integrate the volume of the sphere by collecting values from every lattice point throughout the unit hypercube. A value is unity if the point lies within the hypersphere $r^2 \leq 1$; otherwise it is zero. We sum all values (0 or 1) from every lattice point and obtain an integer equal to the number of lattice points S inside the hypersphere. The total number of lattice points is equal to M^4. That sum corresponds to the total volume of the hypercube, which is 1. Therefore the discrete estimate of the volume of $1/16$th of the hypersphere is S/M^4. We can compare this numerical integration with the analytic value and express the fractional error as the numerical value

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

divided by the analytic value, minus one:

$$\text{Fractional error} = \left| \frac{S/M^4}{\pi^2/32} - 1 \right| .$$

Monte Carlo integration works essentially exactly the same except that the points we choose are not a regular lattice, but rather they are random. Each one is found by taking four new uniform-variate values (between 0 and 1) for the four coordinate values x_i. The point contributes unity if it has $r^2 \leq 1$ and zero otherwise. We obtain a different count S_r. We'll choose to use a total number N of random point positions exactly equal to the number of lattice points $N = M^4$, although we could have made N any integer we liked. The Monte Carlo integration estimate of the volume is S_r/N.

I wrote a computer code to carry out these simple procedures, and compare the fractional errors for values of M ranging from 1 to 100. The results are shown in Fig. 11.4.

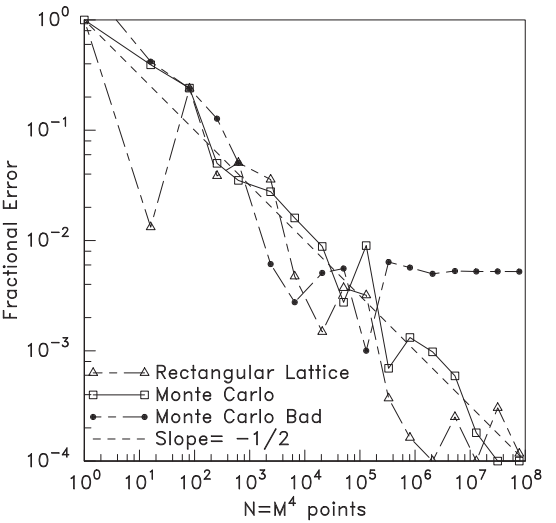

Figure 11.4 Comparing error in the volume of a hypersphere found numerically using lattice and Monte Carlo integration. It turns out that Monte Carlo integration actually does *not* converge significantly faster than lattice integration, contrary to common wisdom. They both converge approximately like $1/\sqrt{N}$ (logarithmic slope $= -1/2$). What's more, if one uses a "bad" random-number generator (the Monte Carlo Bad line) it is possible that the random integration will cease converging at some number, because it gives only a finite length of independent random numbers, which in this case is exhausted at roughly a million.

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press

Four-dimensional lattice integration does as well as Monte Carlo for this sphere. Lattice integration is not as bad as the dubious assumption of fractional uncertainty $1/M = N^{-1/d}$ suggests; it is more like $N^{-2/d}$ for $d > 1$. Only at higher dimensionality than $d = 4$ do tests show the advantages of Monte Carlo integration beginning to be significant. As a bonus, this integration experiment detects poor random-number generators.[12]

Exercise 11. Monte Carlo techniques

1. A random variable is required, distributed on the interval $0 \le x \le \infty$ with probability distribution $p(x) = k\exp(-kx)$, with k a constant. A library routine is available that returns a uniform random variate y (i.e. with uniform probability $0 \le y \le 1$). Give formulas and an algorithm to obtain the required randomly distributed x value from the returned y value.
2. Particles that have a Maxwellian distribution

$$f(\boldsymbol{v}) = n\left(\frac{m}{2\pi kT}\right)^{3/2}\exp\left(-\frac{mv^2}{2kT}\right) \tag{11.17}$$

   cross a boundary into a simulation region.

   (a) Find the cumulative probability distribution $P(v_n)$ that ought to be used to determine the velocity v_n normal to the boundary; of the injected particles.
   (b) What is the total rate of injection per unit area that should be used?
   (c) If the timestep duration is Δt, and the total rate of crossing a face is r such that $r\Delta t = 1$, what probability distribution should be used to determine statistically the *actual* integer number of particles injected at each step?
   (d) What should be the probability of 0, 1, or 2 injections?
3. Write a code to perform a Monto Carlo integration of the area under the curve $y = (1 - x^2)^{0.3}$ for the interval $-1 \le x \le 1$. Experiment with different total numbers of sample points, and determine the area accurate to 0.2%, and approximately how many sample points are needed for that accuracy.

[12] The random generators I used here are both portable. The good generator is the `RANLUX` routine described by M. Luscher, (1994), *Computer Physics Communications* **79** 100, and F. James, (1994), *Computer Physics Communications* **79** 111, used with the lowest luxury level. The bad generator is the `RAN1` routine from the first (FORTRAN) edition of *Numerical Recipes*. It was replaced by better routines in later editions.

https://doi.org/10.1017/CBO9781316155516.012 Published online by Cambridge University Press