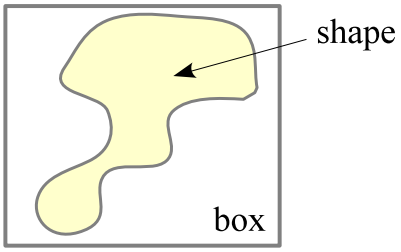
12 Monte Carlo Techniques

12.1 Monte Carlo Integration

So-called Monte Carlo integrations lie at the heart of many stochastic simulation methods. The basic idea can be intuitively understood with the "dartboard method" of integrating the area of an irregular domain.



This works as follows:

- Choose points randomly (i.e. uniformly) within the box.
- We know that the probability that a point hits inside the area is proportional to the ratio of the areas:

$$p(\text{dart hits inside area}) = \frac{A_{\text{shape}}}{A_{\text{box}}} \tag{12.1}$$

- We can now approximate this probability, and hence the area ratio, through the experimental result:

$$\frac{A_{\text{shape}}}{A_{\text{box}}} \simeq \frac{\#\text{hits in shape}}{\#\text{hits in box}} \tag{12.2}$$

  This is expected to become arbitrarily accurate as the number of trials goes to infinity.

Standard Monte Carlo integration

Let us now formalize this technique. We consider an integral in d-dimensions,

$$I = \int_V f(\mathbf{x})\, \mathrm{d}^d\mathbf{x}, \tag{12.3}$$

where V is a d-dimensional hypercube with (for simplicity) dimensions $[0,1]^d$. To compute this as a Monte Carlo integral, we do the following:

1. Generate N random vectors $0 \leq \mathbf{x}_i \leq 1$ with flat distribution (i.e. each component of the vector is drawn independently from a uniform distribution).

2. We compute
$$I_N = \frac{V}{N} \sum_i^N f(\mathbf{x}_i). \tag{12.4}$$
For $N \to \infty$, we then get $I_n \to I$.

3. The error of the result scales as $1/\sqrt{N}$, *independent* of the number of dimensions of the integral.

Especially the last point is quite remarkable – we'll later have to look at this in more detail. Before we do this, it is instructive to compare with the steps taken in standard integration techniques. In them, we divide each dimension in n regularly spaced points. The total number of points is hence $N = n^d$. Depending on the integration rule selected, the error will then scale as some power of $1/n$. For example, for the midpoint and trapezoidal rules, it will simply be $\propto 1/n^2$, and for Simpson's rule $\propto 1/n^4$.

If d is small, it is clear that the Monte Carlo integration has much larger errors than standard methods when the same N is used. However, the higher d becomes, the better Monte Carlos looks because then the standard method can spend comparatively fewer regular sampling points per dimension.

One can then for example ask: At what point is Monte Carlo as good as Simpson? Well, Simpson's error should scale as
$$\frac{1}{n^4} = \frac{1}{N^{4/d}}, \tag{12.5}$$
which starts to decline with N more weakly than Monte Carlo integration when $d \geq 8$. We hence clearly see that high dimensional problems are the regime where Monte Carlo integration becomes particularly interesting.

In fact, in some lattice simulations one has dimensions in the range $d = 10^6 - 10^{10}$. Here the only viable approach is to use Monte Carlo integration. In practice, standard methods fail already at much more moderate numbers of dimensions. For example, even with $d = 10$, putting down just a grid with $n = 10$ cells per dimension already yields a number of $N = 10^{10}$ grid points.

12.2 Error in Monte Carlo integration

Let $y_i = V f(\mathbf{x}_i)$ be the value of the i-th function evaluation of our Monte Carlo integration. After N samples, we can thus write the approximation to the desired integral as
$$I_N = \frac{y_1 + y_2 + \ldots + y_N}{N} \tag{12.6}$$

The error of this quantity is *defined* as the width of its probability distribution $P_N(I_N)$, i.e.

$$\sigma_N^2 \equiv \left\langle I_N^2 \right\rangle - \langle I_N \rangle^2 . \tag{12.7}$$

Let's try to calculate this in order to get an idea of the size of this error. First, let's introduce the probability distribution of the y_i, denoted with $p(y)$. For it we have

$$\int p(y)\,\mathrm{d}y = 1, \tag{12.8}$$

$$\langle y \rangle = \int y\,p(y)\,\mathrm{d}y, \tag{12.9}$$

$$\left\langle y^2 \right\rangle = \int y^2\,p(y)\,\mathrm{d}y, \tag{12.10}$$

$$\sigma^2 = \left\langle y^2 \right\rangle - \langle y \rangle^2 . \tag{12.11}$$

We can then write

$$P_N(I_N) = \int \delta\left(I_N - \sum_i \frac{y_i}{N}\right) p(y_1)\,p(y_2)\cdots p(y_N)\,\mathrm{d}y_1\,\mathrm{d}y_1\cdots\mathrm{d}y_N, \tag{12.12}$$

where the Dirac delta-function enforces that the average of the y_i is equal to I_N. We now take the Fourier transform of $p(y)$:

$$\hat{p}(k) = \int p(y)\,\mathrm{e}^{ik(y-\langle y\rangle)}\,\mathrm{d}y \tag{12.13}$$

Similarly, let's consider the Fourier transform of $P_N(I_N)$:

$$\begin{aligned}
\hat{P}_N(k) &= \int P_N(I_N)\,\mathrm{e}^{ik(I_N-\langle I_N\rangle)}\,\mathrm{d}I_N && (12.14)\\
&= \int p(y_1)\cdots p(y_N)\,\mathrm{e}^{i\frac{k}{N}(y_1-\langle y_1\rangle+y_2-\langle y_2\rangle+\ldots+y_N-\langle y_N\rangle)}\mathrm{d}y_1\cdots\mathrm{d}y_N && (12.15)\\
&= \left[\hat{p}\left(\frac{k}{N}\right)\right]^N . && (12.16)
\end{aligned}$$

Here we made use of $\langle I_N \rangle = \langle y \rangle$. Now we expand $\hat{p}\left(\frac{k}{N}\right)$ in powers of k/N, in the limit of large N. We get

$$\begin{aligned}
\hat{p}\left(\frac{k}{N}\right) &= \int p(p)\,\mathrm{e}^{i\frac{k}{N}(y-\langle y\rangle)}\,\mathrm{d}y && (12.17)\\
&= \int p(y)\left[1 + \frac{ik}{N}(y-\langle y\rangle) - \frac{k^2}{2N^2}(y-\langle y\rangle)^2 + \ldots\right]\mathrm{d}y && (12.18)\\
&= 1 - \frac{k^2\sigma^2}{2N^2} + \ldots && (12.19)
\end{aligned}$$

Thus we find

$$\hat{P}_N(k) = \left[\hat{p}\left(\frac{k}{N}\right)\right]^N = \left(1 - \frac{k^2\sigma^2}{2N^2}\right)^N \simeq 1 - \frac{k^2\sigma^2}{2N} \simeq \mathrm{e}^{-\frac{k^2\sigma^2}{2N}}, \tag{12.20}$$

because N is very large. With this result in hand, we can now use it to calculate $P_N(I_N)$ through an inverse Fourier transform:

$$\begin{aligned} P_N(I_N) &= \frac{1}{2\pi}\int \mathrm{e}^{-ik(I_N - \langle I_N\rangle)}\,\hat{P}_N(k)\,\mathrm{d}k & (12.21)\\ &= \frac{1}{2\pi}\int \mathrm{e}^{-\frac{k^2\sigma^2}{2N} - ik(I_N - \langle I_N\rangle)}\,\mathrm{d}k & (12.22)\\ &= \frac{1}{2\pi}\mathrm{e}^{-\frac{1}{2}\frac{N}{\sigma^2}(I_N - \langle y\rangle)^2}\int \mathrm{e}^{-\frac{\sigma^2}{2N}\left(k + i(I_N - \langle y\rangle)\frac{N}{\sigma^2}\right)^2}\,\mathrm{d}k & (12.23) \end{aligned}$$

We now can do this integral by shifting k and recalling

$$\int \exp(-\alpha x^2)\,\mathrm{d}x = \sqrt{\frac{\pi}{\alpha}}, \tag{12.24}$$

obtaining

$$P_N(I_N) = \frac{\sqrt{N}}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\frac{N}{\sigma^2}(I_N - \langle y\rangle)^2\right). \tag{12.25}$$

This is a Gaussian with dispersion $\sigma_N = \sigma/\sqrt{N}$, *independent* of the shape of $p(y)$. What we just have derived is the *central limit theorem*! Independent of the detailed shape of $p(y)$, if we average enough of these distributions we will get a Gaussian.

Summary:

For standard Monte Carlo integration with N samples, the error is

$$\sigma_N = V\sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{N}} \tag{12.26}$$

where $\langle f\rangle$ and $\langle f^2\rangle$ are exact moments of the function we integrate, i.e.

$$\langle f\rangle \equiv \frac{1}{V}\int f(x)\mathrm{d}x = \frac{1}{V}\int y\,p(y)\mathrm{d}y. \tag{12.27}$$

In practice, we can estimate these moments from the Monte-Carlo samples themselves, i.e. we can estimate

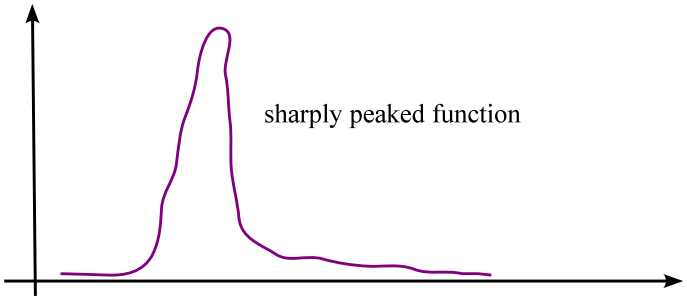
$$\langle f\rangle \simeq \frac{1}{N}\sum_i f(x_i), \tag{12.28}$$

$$\left\langle f^2\right\rangle \simeq \frac{1}{N}\sum_i f^2(x_i), \tag{12.29}$$

and then use these moments to estimate the error.

12.3 Importance Sampling

One common problem in Monte Carlo integration is that often the integrand is very small on a dominant fraction of the integration volume. For example, if the integrand is sharply peaked, only points sampled close to the peak will give a significant contribution.



The idea of **importance sampling** is to choose the random points somehow preferentially around the peak, and putting less points where the integrand is small. This should be more efficient and help to reduce the error for a given number of points.

So let us assume we want to integrate

$$I = \int_V f(x)\,\mathrm{d}x, \tag{12.30}$$

and suppose we choose a distribution $p(x)$ which is close to the function $f(x)$, but which is simple enough so that it is possible to generate x-values from this distribution. We can then write:

$$I = \int_V p(x)\frac{f(x)}{p(x)}\,\mathrm{d}x. \tag{12.31}$$

Thus, if we sample points around a point x with probability $\mathrm{d}p = p(x)\,\mathrm{d}x$ (which is exactly the definition of sampling from the distribution $p(x)$), we simply obtain:

$$I = \lim_{N\to\infty}\frac{1}{N}\sum_i \frac{f(x_i)}{p(x_i)}. \tag{12.32}$$

Because f/p is flatter than f if the shape of p is similar to that of f, the variance of f/p will be smaller than the variance of f, i.e. we obtain a smaller error for given N. The ideal choice is $p(x) \propto f(x)$. This is often not possible in practice, but in fact possible in lattice Monte Carlo simulations, as we will see.

Example for importance sampling

Let's consider a simple 1D integration to demonstrate the concept of importance sampling. The integral we want to compute is:

$$I = \int_0^1 \left(x^{-1/3} + \frac{x}{10}\right)\,\mathrm{d}x. \tag{12.33}$$

This can be solved analytically and has the value $I = 31/20 \simeq 1.55$, so we don't really need Monte Carlo integration here, but for the sake of demonstrating the method we apply it anyway.

Doing this integral with standard Monte Carlo integration gives the error

$$\sigma_N = \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{N}} \simeq \frac{0.85}{\sqrt{N}}. \tag{12.34}$$

Let's now try to do it with importance sampling, using the sampling probability

$$p(x) = \frac{2}{3} x^{-1/3} \tag{12.35}$$

over the interval $0 < x \leq 1$, based on the realization that $p(x)$ captures part of the shape of $f(x)$, while being simple enough to allow a creation of properly sampled points by direct inversion (see below). The new function to integrate is then $g = f/p$, and the width of the corresponding Monte Carlo error distribution function becomes

$$\sigma_N = \sqrt{\frac{\langle g^2 \rangle - \langle g \rangle^2}{N}} \simeq \frac{0.045}{\sqrt{N}}. \tag{12.36}$$

This is nearly 20 times better than obtained with plain sampling, hence a substantial gain in efficiency has been reached.

12.4 Random number generation

Good random number obviously play a central role in Monte Carlo techniques.

- Usually they are produced by *deterministic algorithms* leading to *pseudorandom numbers*.
- Such pseudorandom numbers need to be distinguished from "truly random" numbers generated by some physical process (like rolling the dice, radioactive decay, quantum transitions, etc.). Some modern CPUs include a hardware random number generator, based for example on coupled non-linear oscillators and additional sources of entropy. However, these generators are normally not used for Monte Carlo techniques, because
  - the sequence is not repeatable, making debugging difficult and preventing exact reproducibility
  - the generators are often slow
  - the quality of the distribution may not be perfect
  - the quality of the distribution may degrade with time, or correlate in subtle ways with environmental factors such as operating temperature, etc.

- There is value in having good random numbers. In 1950, the RAND corporation published a book entitled "1 million random digits", whose primary virtue is to contain no discernable information at all. This classic is available online (`http://www.rand.org/publications/classics/randomdigits`).

12.4.1 Pseudo-random numbers

Here we usually create an integer sequence that is then converted to a floating point number in the interval $[0, 1[$. The essential desirable properties of a good random number generator are:

- Repeatability: For the same seed, we want to obtain the same sequence of random numbers.
- Randomness: Good random numbers should
  - be uniformly and homogeneously distributed in the interval $[0, 1[$.
  - be independent of each other, i.e. show no correlations whatsoever (this is difficult and not true exactly for pseudo-random number generators)
- Speed: In modern applications, we may need billions of random numbers.
- Portability: We want the same results on different computer architectures.
- Long period: After a finite number of pseudo-random numbers, the sequence repeats. This period should be as large as possible.
- Insensitivity to seed: Neither the period nor the quality of the randomness should depend on the value of the seed, i.e. on where the sequence is started.

Linear congruential generators

The simplest pseudo-random number generators work with an integer mapping of the form

$$X_{\mathrm{i}+1} = (aX_i + b) \mod m, \tag{12.37}$$

where a, b, and m are integers. The numbers X_i lie then in the range $[0, m-1]$ and can be mapped to floating point numbers in the unit interval by dividing with m. It is clear that such a generator can have a period of at most m. Examples for such *linear congruential generators* include:

- **ANSI-C:**

$$a = 1103515245 \quad b = 12345 \quad m = 2^{31} \tag{12.38}$$

  The period here is quite small, just $m = 2^{31} \sim 2 \times 10^9$, which is quickly reached in modern computers. This is not good enough for serious Monte Carlo applications.

- **RAND generator in Matlab:**

$$a = 16807 \quad b = 0 \quad m = 2^{31} - 1 \tag{12.39}$$

  Again, this has an uncomfortably short period.

- **UNIX drand48():**

$$a = 25214903917 \quad b = 11 \quad m = 2^{48} \tag{12.40}$$

  This is starting to be somewhat usable, with a period of $2^{48} \simeq 2.8 \times 10^{14}$. Note however that the low order bits have shorter cycling times and show less randomness than they should, which is a common problem for all simple linear congruential random number generators.

- **NAG-generator:**

$$a = 13^{13} \quad b = 0 \quad m = 2^{59} \tag{12.41}$$

  This has a very long period, but the low order bits are still not very good.

To get better random numbers, one needs to go to more complicated schemes than a simple integer mapping. One approach is to combine two or several linear congruential mappings. This is done for example in **ran2** of Numerical Recipes. This uses

$$X_{i+1} = (40014 X_i) \mod 2147483563 \tag{12.42}$$
$$Y_{i+1} = (40692 Y_i) \mod 2147483399 \tag{12.43}$$
$$Z_{i+1} = (X_i + Y_i) \mod 2147483563 \tag{12.44}$$

The Z_i are then mapped to floating point numbers. Here the period is extended to $\sim 10^{18}$.

Lagged Fibonacci generators

A refinement of this approach consists of using several integers to define the internal state of the generator. One then uses a prescription of the form

$$X_i = (X_{i-p} \odot X_{i-q}) \mod m \tag{12.45}$$

to create new integers, where p and q are the 'lags' (offsets to other past numbers), and $\odot$ is some arithmetic operation, for example addition, multiplication, etc., or also bitwise logical operations such as XOR. For large lags, the quality of these generators becomes very good.

For example, **RANLUX** is of this type, reaching a period 10^{171}. Another modern generator using this principle is the **Mersenne Twister**, also known as MT19937. This uses 624 internal 32-bit integers to describe its state, and abundantly employs XOR as well as other bit-shuffling and swapping operations. It's period is huge, a staggering $2^{19937} - 1$. This should be a pretty good choice for Monte Carlo! The Mersenne Twister is for example available in the GSL-library.[1]

[1] `http://www.gnu.org/software/gsl`

12.5 Using random numbers

Random numbers are usually generated in the standard interval $[0, 1[$ with a uniform distribution. If that's what you need – fine. But often we need random numbers drawn from some other probability distribution function (e.g. a Gaussian). How is this done?

12.5.1 Exact inversion

Recall, the PDF satisfies $\int p(x)\,\mathrm{d}x = 1$ and $p(x) \geq 0$ for all x. Such probability distributions can be transformed to other distributions by observing conservation of probability:

$$p_1(x)\,\mathrm{d}x = p_2(y)\,\mathrm{d}y \tag{12.46}$$

where $y = y(x)$. This leads to the transformation rule

$$p_2(y) = p_1(x)\left|\frac{\mathrm{d}y}{\mathrm{d}x}\right|, \tag{12.47}$$

where here a modulus has been added to neutralize a possible sign change due to the mapping. This can now be used as follows: Suppose we know $p_1(x)$ (usually the distribution returned by our random number generator, in which case $p_1(x) = 1$) and we have a desired distribution $p_2(y)$, then we need to find the mapping $y = y(x)$ that transforms one into the other. We can obtain this by integrating the differential equation

$$\int_{-\infty}^{x} p_1(x')\,\mathrm{d}x' = \int_{-\infty}^{y} p_2(y')\,\mathrm{d}y'. \tag{12.48}$$

This is simply saying that $P_1(x) = P_2(y)$, where $P_1(x)$ and $P_2(y)$ are the cumulative probability distribution functions of p_1 and p_2, respectively. Hence, we need to calculate

$$y = P_2^{-1}\left[P_1(x)\right]. \tag{12.49}$$

In case $p_1(x)$ is an ordinary random number generator, this can also be written as

$$x = \int_{-\infty}^{y} p_2(y')\,\mathrm{d}y'. \tag{12.50}$$

Unfortunately, the inversion cannot always be carried out algebraically, but if this is possible, this is the method of choice.

Example

Suppose you want to have random numbers from the distribution

$$p(y) = \frac{1}{4}y^3 \quad \text{for } y \in [0, 2]. \tag{12.51}$$

The cumulative distribution is here

$$P(y) = \int_0^y \frac{y'^3}{4}\,\mathrm{d}y' = \frac{y^4}{16}. \tag{12.52}$$

Hence, we can draw random numbers uniformly from $x \in [0, 1[$ and concert them to

$$y = (16x)^{1/4}, \tag{12.53}$$

which then sample our desired distribution function.

Important special cases

The Gaussian distribution

$$p(y) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y^2}{2}\right) \tag{12.54}$$

is often needed. The cumulative distribution is the error function, which can not be easily inverted without resorting to iterative (and hence comparatively expensive) methods.

There is however a simple trick, known as the Box-Muller method, that can circumvent this issue. Suppose we consider generating a 2D Gaussian distribution

$$p(x, y) = \frac{1}{2\pi} \exp\left(-\frac{x^2 + y^2}{2}\right), \tag{12.55}$$

which is simply the product of two 1D-distributions. We can transform this to polar coordinates in the (x, y)-plane:

$$p(x, y)\,\mathrm{d}x\,\mathrm{d}y = \frac{1}{2\pi} \exp\left(-\frac{r^2}{2}\right) r\,\mathrm{d}r\,\mathrm{d}\phi. \tag{12.56}$$

Hence ϕ is uniformly distributed in $[0, 2\pi[$, i.e.

$$\phi = 2\pi \cdot X_1 \tag{12.57}$$

for some standard random number X_1 from the unit interval. For the radial coordinate we have on the other hand:

$$X_2 = \int_0^r r' \exp\left(-\frac{r'^2}{2}\right) \mathrm{d}r' \tag{12.58}$$

This can be integrated and inverted to yield

$$r = \sqrt{-2 \ln X_2}, \tag{12.59}$$

where X_2 is again a random number independently drawn from $[0, 1[$. Finally, we can calculate

$$x = r \cos\phi, \tag{12.60}$$

$$y = r \sin\phi, \tag{12.61}$$

which now yields two perfectly fine Gaussian distributed numbers x and y, which may both be used. This procedure hence always converts two random numbers from $[0, 1[$ to two independent Gaussian distributed numbers.

12.5.2 Rejection method

Assume that $p(x)$ is the desired random number distribution, and $f(x)$ is the distribution that we can create. If we have

$$p(x) \leq C \cdot f(x) \tag{12.62}$$

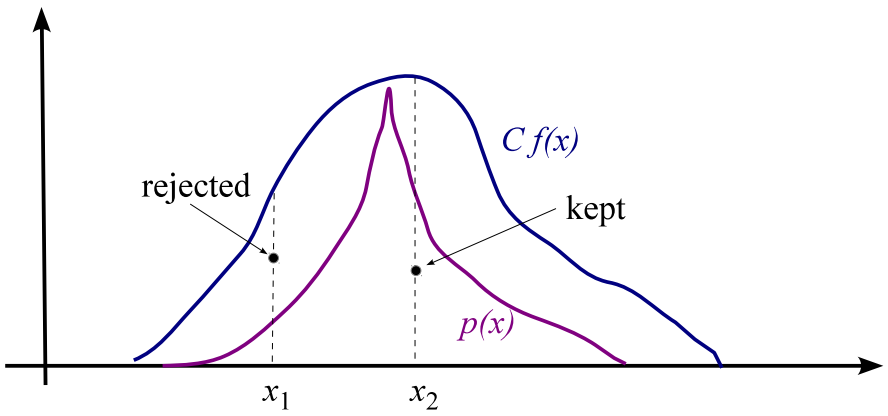
with some known constant C, then we can generate random numbers that sample $p(x)$ with the rejection method. This method works as follows:

1. Generate an x from $f(x)$.
2. Generate a y from a uniform distribution with the bounds $0 \leq y < C \cdot f(x)$.
3. If $y \leq p(x)$ return x as a sample value.
4. Otherwise, i.e. for $y > p(x)$ reject the trial value for x and repeat at step 1.

The probability $\mathrm{d}q$ to get a certain x within $\mathrm{d}x$ with this procedure is:

$$\mathrm{d}q = f(x)\mathrm{d}x \cdot \frac{p(x)}{Cf(x)} = \frac{1}{C}\, p(x)\, \mathrm{d}x \propto p(x)\, \mathrm{d}x. \tag{12.63}$$

Hence this will reproduce the desired probability distribution.



There are a number of advantaged of this approach, in particular, it works in any dimension, and $p(x)$ does not necessarily have to be normalized. The main disadvantage can lie in a low efficiency if the rejection rate is high. The latter is given by the complement to the acceptance rate, which is given by the area under $p(x)$ relative to the area under $Cf(x)$.

Example

Let's assume we want to distribute points uniformly on a sphere. The standard way is to use direct inversion. In 3D, this is still readily possible by the use of spherical polar coordinates. We have for the surface element

$$\sin\theta\, \mathrm{d}\theta\, \mathrm{d}\phi = \mathrm{d}\cos\theta\ \mathrm{d}\phi, \tag{12.64}$$

hence the distributions of $\mathrm{d}\cos\theta$ and $\mathrm{d}\phi$ are uniform over their range. Hence we can set

$$\cos\theta = 2\,u_1 - 1, \tag{12.65}$$

$$\phi = 2\pi u_2, \tag{12.66}$$

where u_1 and u_2 are standard uniform numbers. We can then calculate the coordinates as

$$x = \sin\theta\,\cos\phi, \tag{12.67}$$

$$y = \sin\theta\,\sin\phi, \tag{12.68}$$

$$z = \cos\theta. \tag{12.69}$$

This is fine, but cumbersome to generalize to hyperspheres in higher dimensions. A much simpler approach is to use rejection sampling. Suppose we draw three random numbers u_1, u_2, and u_3, which we then map to $[-1,1]$ through $\tilde{u}_i = 2u_i - 1$. Now we calculate $r^2 = \tilde{u}_1^2 + \tilde{u}_2^2 + \tilde{u}_3^2$, and use the rejection method: We only keep the point if $r^2 \leq 1$, which effectively uniformly samples the inside of a sphere. If we then stretch the kept points as

$$x = \frac{\tilde{u}_1}{r}, \quad y = \frac{\tilde{u}_2}{r}, \quad z = \frac{\tilde{u}_3}{r}, \tag{12.70}$$

they are uniformly distributed on the unit sphere. This method works for any number of dimensions.

12.5.3 Sampling with a stochastic process

There are situations when neither direct inversion nor the rejection method can be readily used to sample from a given distribution function $p(x)$. In this case we can construct a sample of $p(x)$ through a stochastic process that has $p(x)$ as its equilibrium distribution.

We will accomplish this with a so-called *Markov process*, which generates a Markov chain. A Markov chain is a discrete sequence of states,

$$x_1 \xrightarrow{f} x_2 \xrightarrow{f} x_3 \xrightarrow{f} \ldots \xrightarrow{f} x_n, \tag{12.71}$$

where f is a Monte Carlo update operator. The characterizing property of a Markov process is that the transition probability from one state to the next state in the chain,

$$W_f(x \to x') = W_f(x'|x), \tag{12.72}$$

depends *only* on the current state, i.e. information about the history is not used at all. Note that f can here mediate a small update or an arbitrarily large one.

The transition probability has the natural properties

$$\int W_f(x \to x')\,\mathrm{d}x' = 1, \tag{12.73}$$

and $W_f(x \to x') \geq 0$.

We can also apply the transition probability to whole probability distributions, getting the new probability distribution after one transition:

$$p(x) \xrightarrow{f} p'(x') = \int p(x) W_f(x \to x') \, \mathrm{d}x. \tag{12.74}$$

We will now demand two properties of the update step that will turn the Markov process into a very powerful tool:

1. f must preserve $p_{\mathrm{eq}}(x)$ as an equilibrium distribution of the stochastic process, or in other words $p_{\mathrm{eq}}(x)$ must be a fix point of f. This requires

$$p_{\mathrm{eq}}(x') = \int p_{\mathrm{eq}}(x) W_f(x \to x') \, \mathrm{d}x. \tag{12.75}$$

2. Starting from any state x, repeated applications of f must be able to get arbitrarily close to any other state x'. This is called the ergodic property.

Two important results follow from these properties:

- Any ensemble of states approaches the equilibrium distribution if f is applied sufficiently often.
- The collection of states in a single Markov chain under the action of f approaches $p(x)$ as the number of steps goes to infinity.

Let us proof the first of these results. To this end, let $p(x)$ be the PDF of the initial ensemble, and $p_{\mathrm{eq}}(x)$ the equilibrium distribution. After applying f once, we obtain $p'(x')$. We now want to show that p' is closer to p_{eq} than p. To this end, we consider the norm

$$\begin{aligned}
||p' - p_{\mathrm{eq}}|| &\equiv \int |p'(x') - p_{\mathrm{eq}}(x')| \, \mathrm{d}x' && (12.76)\\
&= \int \mathrm{d}x' \left| \int \mathrm{d}x \, W_f(x \to x')(p(x) - p_{\mathrm{eq}}(x)) \right| && (12.77)\\
&\leq \int \mathrm{d}x' \int \mathrm{d}x \, W_f(x \to x') \, |p(x) - p_{\mathrm{eq}}(x)| && (12.78)\\
&= \int \mathrm{d}x \, |p(x) - p_{\mathrm{eq}}(x)| && (12.79)\\
&= ||p - p_{\mathrm{eq}}||. && (12.80)
\end{aligned}$$

For the third line, we have basically used the triangle inequality, $|a + b| \leq |a| + |b|$. Thus, the difference between p and p_{eq} shrinks if f is applied. But, perhaps $||p - p_{\mathrm{eq}}||$ gets stuck at some finite value and doesn't really go to zero. This would mean that there must be another fix-point $\tilde{p}_{\mathrm{eq}}$ with $||\tilde{p}_{\mathrm{eq}} - p_{\mathrm{eq}}|| > 0$, and $||\tilde{p}'_{\mathrm{eq}} - p'_{\mathrm{eq}}|| = ||\tilde{p}_{\mathrm{eq}} - p_{\mathrm{eq}}||$.

However, the ergodicity property of the mapping f implies that there are some x' for which

$$\left| \int \mathrm{d}x \, W_f(x \to x')(\tilde{p}_{\mathrm{eq}}(x) - p_{\mathrm{eq}}(x)) \right| < \int \mathrm{d}x \, W_f(x \to x') \, |\tilde{p}_{\mathrm{eq}}(x) - p_{\mathrm{eq}}(x)| \, . \quad (12.81)$$

This is because if A is the set for which $\tilde{p}_{\mathrm{eq}}(x) - p_{\mathrm{eq}}(x) \leq 0$, and B the set for which $\tilde{p}_{\mathrm{eq}}(x) - p_{\mathrm{eq}}(x) > 0$, then there must be some x' from B and some x from A for which we have a non-zero $W_f(x \to x') > 0$, otherwise the two sets would be isolated from each other, violating the ergodic assumption. On the other hand, for the norm of $||\tilde{p}'_{\mathrm{eq}} - p'_{\mathrm{eq}}||$ we get

$$\begin{aligned} ||\tilde{p}'_{\mathrm{eq}} - p'_{\mathrm{eq}}|| &= \int \mathrm{d}x' \, |\tilde{p}'(x')_{\mathrm{eq}} - p'_{\mathrm{eq}}(x')| = \int \mathrm{d}x' \left| \int \mathrm{d}x \, W_f(x \to x')(\tilde{p}(x)_{\mathrm{eq}} - p_{\mathrm{eq}}(x)) \right| \\ &< \int \mathrm{d}x' \int \mathrm{d}x \, W_f(x \to x') \, |\tilde{p}(x)_{\mathrm{eq}} - p_{\mathrm{eq}}(x)| \quad (12.82) \\ &= \int \mathrm{d}x \, |\tilde{p}(x)_{\mathrm{eq}} - p_{\mathrm{eq}}(x)| = ||\tilde{p}_{\mathrm{eq}} - p_{\mathrm{eq}}||, \quad (12.83) \end{aligned}$$

where for establishing the $<$-sign we used equation (12.81). The conclusion reached here, $||\tilde{p}'_{\mathrm{eq}} - p'_{\mathrm{eq}}|| < ||\tilde{p}_{\mathrm{eq}} - p_{\mathrm{eq}}||$, contradicts the existence of two equilibrium distributions.

Detailed balance

Almost all of the commonly used update steps follow the **detailed balance condition**, i.e.:

$$p_{\mathrm{eq}}(x) \cdot W_f(x \to x') = p_{\mathrm{eq}}(x') \cdot W_f(x' \to x). \quad (12.84)$$

Here it is obvious and easy to show that $p_{\mathrm{eq}}(x)$ is a fix point under f, while for other choices of f this may still be the case but could be difficult to prove.

So detailed balance and ergodicity are already sufficient conditions to obtain a Markov chain that samples $p_{\mathrm{eq}}(x)$. But we still need to find a concrete realization of W_f.

12.5.4 The Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm provides a simple and generic way for constructing a suitable transition operation. It works as follows:

1. When the currents state is x, propose a new state x' with a proposal probability $q(x \to x')$.

2. Calculate the Hasting's ratio

$$r = \min \left(1, \frac{p(x') \, q(x' \to x)}{p(x) \, q(x \to x')} \right), \quad (12.85)$$

   where the min-operation is used to restrict the value of r to the range $[0, 1]$.

3. Accept the proposed move with probability r (i.e. draw a random number $u \in [0, 1[$, and if its smaller than r, accept), in which case x' is made the next element in the Markov chain. Otherwise, the proposed state is *rejected*, and the old state is added (again) as an element in the Markov chain. This is sometimes called the Metropolis rejection step.

Does the Metropolis algorithm fulfill detailed balance? We can check this by working out the transition probability $W(x \to x')$, which is the product of the proposal probability of the new state and its acceptance probability:

$$W(x \to x') = q(x \to x') \cdot \frac{p(x')\, q(x' \to x)}{p(x)\, q(x \to x')} = \frac{p(x')}{p(x)}\, q(x' \to x). \tag{12.86}$$

Here we have assumed without loss of generality that the Hastings ratio is less than 1. In this case, the inverse transition is then simply given as

$$W(x' \to x) = q(x' \to x) \cdot 1 \tag{12.87}$$

Combining equations (12.86) and (12.87) we then verify the condition of detailed balance.

The proposal probability $q(x \to x')$ is fairly arbitrary – it only must be ergodic, i.e. all states must be reachable through successive applications of q, then the Monte Carlo Markov Chain (MCMC) created by the algorithm will eventually produce a fair sample of the target distribution function $p(x)$. This is a quite remarkable property.

Metropolis update

This is the special case in which the stochastic proposal operator is symmetric, i.e.

$$q(x \to x') = q(x' \to x). \tag{12.88}$$

Then the acceptance probability simply becomes

$$r = \min\left(1, \frac{p(x')}{p(x)}\right). \tag{12.89}$$

A proposed move to a state of higher probability is hence always accepted. (But one sometimes also moves to a proposed state of lower probability.)

The simplest form of such a symmetric update would be something like

$$q(x \to x') : \quad x' = x + e, \tag{12.90}$$

where e is distributed symmetrically around zero and is independent of x. For example, e could be drawn from a normal or uniform distribution of some prescribed width.

Example

Let's come back to our starting point, the generation of a sample from a distribution $p(x)$ with a stochastic process. For simplicity, we want to try this out on a simple Gaussian distribution, $p(x) \propto \exp(-x^2/2)$, using the Metropolis algorithm. As a proposal function, we could for example choose $q(x'|x) : \quad x' = x + (2u - 1)/10$, where u is drawn uniformly from $[0, 1[$. Then we could proceed like this:

1. Start with some x with $p(x) > 0$.
2. Draw u and calculate the proposal x'.
3. Now compute $r = \min\left[1, \exp(-x'^2/2)/\exp(-x^2/2)\right]$.
4. Draw another random number u' from $[0, 1[$ and take x' as new point in the chain if $u' \leq r$, otherwise take x as new point.
5. Repeat at step 2 until you believe you have enough points in the chain.

Note that there can be lots of repeated entries in the chain if the rejection rate is high. Also, you may not use the chain as a randomly sampled sequence from the underlying distribution. Subsequent entries in the change will be highly correlated with each other! Nevertheless, the collection of all the points in a single chain represent a proper sample from the distribution in the limit of an infinitely long chain, thanks to the ergodic property.

12.6 Monte Carlo simulations of lattice models

An important application of MCMC techniques lies in the thermodynamics of physical systems, for example solids described on a lattice. In this case, we may have a field $\phi_{\mathbf{x}}$ at each lattice site $\mathbf{x}$, whose dynamics is described by some Hamiltonian $H(\phi)$.

Assuming the canonical ensemble, the partition function is then given by

$$Z = \int \exp\left[-\frac{H(\phi)}{kT}\right] [\mathrm{d}\phi], \tag{12.91}$$

where $[\mathrm{d}\phi] = \mathrm{d}\phi_1\, \mathrm{d}\phi_2\, \mathrm{d}\phi_3 \cdots \mathrm{d}\phi_N$ is a short-hand notation for the differential volume element in the extremely high-dimensional space of all possible field configurations.

In practice, very often the task to compute the thermal average of some quantity A arises. This is given by

$$\langle A \rangle = \frac{1}{Z} \int A(\phi) \exp\left[-\frac{H(\phi)}{kT}\right] [\mathrm{d}\phi]. \tag{12.92}$$

Unfortunately, because of the high dimensionality, these integrals cannot be carried out with standard techniques. We hence would like to employ Monte Carlo integration combined with importance sampling in which the phase-space points of the

system are chosen according to $p(\phi) \propto \exp\left[-\frac{H(\phi)}{kT}\right]$. For obtaining such a sample, we need a stochastic process, because neither direct inversion nor the rejection method are feasible.

For producing the required Markov chain, one can for example use the Metropolis-Hastings algorithm. If we symmetrically propose new states, then the acceptance probability will become

$$r = \min\left[1, \exp\left(-\frac{H(\phi') - H(\phi)}{kT}\right)\right], \tag{12.93}$$

still leaving many ways for how the proposals are generated.

Another possibility is to employ the so-called *heat bath*, or Gibbs sampler. This directly sets

$$W_f(\phi \to \phi') = C \exp\left(-\frac{H(\phi')}{kT}\right), \tag{12.94}$$

which even doesn't depend on the state ϕ at all. In practice, it is however not always trivial to invert this and actually use it.

In either case, once a sufficiently long Markov chain with sampled states has been constructed, we can use it to straightforwardly calculate alls sorts of thermal averages by replacing the integrals with averages of $A(\phi)$ at the sampled points.

Example: Ising Model

The Ising model is the simplest discrete spin model in which the field variable is $s_{\mathbf{x}} = \pm 1$, i.e. at each lattice site the spin either points up or down. The partition function of the system is

$$Z = \sum_{\{s_{\mathbf{x}}\}} \exp\left[-\beta\left(\frac{1}{2}\sum_{<\mathbf{x},\mathbf{y}>}(1 - s_{\mathbf{x}}s_{\mathbf{y}}) + B\sum_{\mathbf{x}} s_{\mathbf{x}}\right)\right]. \tag{12.95}$$

Here the first sum is over all possible spin configurations. The sum $< \mathbf{x}, \mathbf{y} >$ is only over pairs of nearest lattice sites; only their spin interaction is counted. Finally, B describes an external magnetic field, which one may also put to zero. $\beta = 1/T$ measures the temperature (we use a natural system of units here).

For $B = 0$, the Ising model shows 2nd-order phase transitions if the number of dimensions is larger than one. Then, below a certain critical temperature, spontaneous magnetization of the medium occurs. For $d = 2$, the transition temperature has been calculated analytically by Onsager, $\beta_c = \ln(1 + \sqrt{2})$, but for higher dimensions, analytic solutions are not known. Here one therefore needs to turn to Monte Carlo simulations.

The simplest approach is to use the Metropolis algorithm in which one selects a single spin $s_{\mathbf{x}}$ at lattice site $\mathbf{x}$ and proposes the opposite spin direction $s'_{\mathbf{x}}$. The selection of the lattice site can be done randomly, or in red-black ordering (or even type-writer ordering, but this is less efficient). One then computes the local energy

functional $E_{\mathbf{x}}$ which involves all the terms in the interaction energy that involve the chosen spin. This gives rise to a change $\delta E_{\mathbf{x}} = E_{\mathbf{x}}(s'_{\mathbf{x}}) - E_{\mathbf{x}}(s_{\mathbf{x}})$ due to the spin flip. The acceptance probability is then given as

$$r = \min\left(1, \exp[-\beta\, \delta E_{\mathbf{x}}]\right), \tag{12.96}$$

and with this one can generate a long MC chain with different states of the system, which will eventually represent thermal equilibrium at the prescribed temperature.

For this simple spin system, one may also use the heat bath update, and choose the new spin direction of the selected site according to

$$p(s_{\mathbf{x}}) = \frac{\mathrm{e}^{-\beta E_{\mathbf{x}}(s_{\mathbf{x}}=+1)}}{\mathrm{e}^{-\beta E_{\mathbf{x}}(s_{\mathbf{x}}=+1)} + \mathrm{e}^{-\beta E_{\mathbf{x}}(s_{\mathbf{x}}=-1)}}. \tag{12.97}$$

Here the inversion is trivially possible; one simply draws a random number u from $[0, 1]$ and picks the $+1$ direction when $u < p(s_{\mathbf{x}})$ and spin equal to -1 otherwise.

Once a very long MCMC chain of the system in thermal equilibrium has been calculated, one can simply compute various thermodynamic quantities of interest by straightforward averaging (which really is Monte Carlo integration with importance sampling). For example:

- Mean magnetization: $M = \frac{1}{V}\sum_i s_i$
- Specific heat: $C_V = \frac{1}{V}\frac{\partial E}{\partial T} = \langle E^2\rangle - \langle E\rangle^2$
- Magnetic susceptibility: $\chi_M = \frac{1}{V}\frac{\partial M}{\partial T} = \langle M^2\rangle - \langle M\rangle^2$

12.7 Monte Carlo Markov Chains in parameter estimation

Another important application of MCMC techniques lies in parameter estimation, in particular in the context of *Baysian data analysis*. Often in physics, we want to use some experimental data

$$\mathbf{z} = (z_1, z_2, \ldots, z_n) \tag{12.98}$$

to infer a number of parameters

$$\theta = (\theta_1, \theta_2, \ldots, \theta_m) \tag{12.99}$$

which describe a physical model/theory. For example, observations of the microwave background radiation (where $\mathbf{z}$ might refer to the pixels with measured temperature fluctuations) are used to estimate parameters such as the mean density and expansion rate of the Universe.

One way of constraining the parameters is to consider the likelihood function

$$p(\mathbf{z}|\theta). \tag{12.100}$$

Here one considers the probability for observing the data given the parameters have certain values. If we have $p(\mathbf{z}|\theta') > p(\mathbf{z}|\theta)$ for two parameter sets θ' and θ, then it is natural to assume that θ' is somehow more plausible. This leads to the idea of determining the best guess for the parameters by determining the point with the maximum likelihood.

However, thinking about this for a bit, one realizes that we are actually not interested in the probability of observing the data given some parameters, instead we want the reverse: Because we have the data and the parameters are unknown, we should ask, what statement can we make about the probability distribution function of the (unknown) parameter values given the data? This means that we really would like to have

$$p(\theta|\mathbf{z}). \tag{12.101}$$

To obtain access to this quantity, we can invoke **Bayes theorem**, which simply states

$$p(\theta|\mathbf{z}) = \frac{p(\theta)\, p(\mathbf{z}|\theta)}{p(\mathbf{z})}. \tag{12.102}$$

In the context of Bayesian inferences, the terms in this expression carry names that elucidate their meaning.

- $p(\theta)$ is the so-called *prior*. It encodes the knowledge we already have about θ even before the data is taken. This can be in the form of trivial constraints, such as knowing that a parameter cannot take on negative values, or it can come from previous experiments. If we do know nothing about the values of the parameters, then the prior would be a flat distribution.

- $p(\mathbf{z}|\theta)$ is the likelihood of certain data given the parameters in our theoretical model. This we can normally calculate if we know the experimental errors.

- $p(\theta|\mathbf{z})$ is the *posterior* probability distribution function. It encodes the information we have gained about the parameters given the data $\mathbf{z}$ has been measured/observed. Our goal with the experiment is to map out $p(\theta|\mathbf{z})$. This allows us to give confidence intervals for any of the parameters, and also determines all of their correlations.

- Finally, $p(\mathbf{z})$ is called the *model evidence* and gives the probability distribution function for the data. This is normally not easily accessible. Fortunately, it turns out that this is not problematic as the evidence drops out when the Monte Carlo chain is constructed.

A useful way to think about Bayes theorem as applied here is in terms of information theoretical concepts: It describes how our prior knowledge is updated by new data. This is encoded in the statement:

$$\text{posterior} \propto \text{prior} \times \text{likelihood}. \tag{12.103}$$

Now back to the problem of how we can determine $p(\theta|\mathbf{z})$. We achieve a sampling of this distribution by applying a Monte Carlo Markov Chain that samples the posterior as an equilibrium distribution. To this end, we apply the Metropolis-Hastings algorithm:

1. We adopt a proposal function $q(\theta \to \theta') = q(\theta'|\theta)$ which may only depend on the current point in the chain.

2. We compute the acceptance probability

$$r = \min\left(1, \frac{p(\theta')\, p(\mathbf{z}|\theta')\, q(\theta|\theta')}{p(\theta)\, p(\mathbf{z}|\theta)\, q(\theta'|\theta)}\right) \tag{12.104}$$

   and keep the proposal with this probability, otherwise we use the old point as new point in the chain. We see here that the evidence $p(\mathbf{z})$ drops out in the Hastings ratio, simply because the data does not change. Also, we actually do not need to know the normalizations of prior and likelihood – they also drop out.

Again, once we have produced a sufficiently long chain, it becomes easily possible to calculate integrals involving the posterior as this can be simply be done as a Monte Carlo integral with importance sampling. This allows for example marginalizations over certain parameters in an easy way.

Finally, we should not forget to mention one important caveat of these MCMC techniques. It is usually quite difficult to decide when a chain has reached a sufficient length to safely trust all the obtained results. This is one of the main disadvantages of the MCMC method. While there are some heuristics to tell when 'convergence' has been reached, doing this rigorously is a difficult problem without simple practical solution.