Chapter 4
Metropolis Algorithm

General aspects of the Markov Chain Monte Carlo algorithms, which we learned in the previous chapter, may have left you confused. To really understand the subject, we need concrete examples and implementations. In this chapter, we introduce the Metropolis algorithm [1, 2], which is the most famous version of MCMC, and use it to demonstrate how actual simulations are done.

Though MCMC is used for complicated calculations, when we learn how to use them there is absolutely no need for using complicated examples. In this chapter, we start with the easiest example: univariate integration. This example tells you all the essence of the Markov Chain Monte Carlo algorithms. Toward the end of the chapter, we discuss the multivariate version as well. Hopefully, you will be surprised in a good sense: the generalization to multivariate integral is very straightforward.

4.1 Metropolis Algorithm

Suppose that a probability distribution $P(x)$ is written as

$$P(x) = \frac{e^{-S(x)}}{Z}. \tag{4.1}$$

In physics, the function $S(x)$ is called the action and the normalization factor Z is called the partition function. If you are reading this book having the application to statistics in mind, you can regard S as log-likelihood up to a sign. Below, we assume that $S(x)$ is a continuous function of a real variable x.[1] In the case of the Gaussian

[1] An example with discrete variables is the Ising model which will be explained in Sect. 6.2.

distribution, the action is $S(x) = \frac{x^2}{2}$ and the partition function is $Z = \sqrt{2\pi}$. Usually, in practical applications, only $S(x)$ is known and Z is unknown.

In the Metropolis algorithm, starting with an initial value $x^{(0)}$, a sequence $x^{(1)}$, $x^{(2)}, \ldots, x^{(k)}, x^{(k+1)}, \ldots$ is generated as follows:

Metropolis algorithm

1. Choose a real number Δx randomly, and propose $x' = x^{(k)} + \Delta x$ as a candidate for $x^{(k+1)}$. To satisfy the detailed balance condition, we choose the probability distribution of Δx such that Δx and $-\Delta x$ appear with the same probability. (Here, we choose an appropriate $c > 0$ and use the uniform random numbers between $-c$ and $+c$.)
2. Metropolis test: the candidate x' is accepted and the value of x is updated as $x^{(k+1)} = x'$ with a probability $\min(1, e^{S(x^{(k)})-S(x')})$. (Here $\min(1, e^{S(x^{(k)})-S(x')})$ means the smaller one of 1 and $e^{S(x^{(k)})-S(x')}$.) Otherwise x' is rejected and the value of x remains unchanged, as $x^{(k+1)} = x^{(k)}$.

Note that Δx is chosen randomly at each k. For the Metropolis test, a uniform random number r between 0 and 1 is generated, and the proposal x' is accepted if $r < e^{S(x^{(k)})-S(x')}$.

Among the four conditions listed in Chap. 3, we can immediately check three of them:

- We choose Δx randomly without referring to the history, so it is trivially a Markov chain.
- Because we are considering a connected domain of integration, any pair of x and x' can be connected with a finite number of steps. Hence, this Markov chain is irreducible.
- For any $n_s = 1, 2, \ldots$ and x, there is a path connecting x and itself with n_s steps. ($n_s = 1$ is realized when $\Delta x = 0$. Except for the maxima of $S(x)$, $n_s = 1$ can be realized also when the proposed candidate x' is rejected at the Metropolis test.) Hence, the period is 1 for any x, and this Markov chain is aperiodic.

The detailed balance condition is also satisfied. This is a little bit nontrivial, so let us follow the logic carefully:

- First of all, because we assume $-c < \Delta x < c$, the transition probability is zero if $|x - x'| \geq c$:

$$T(x \to x') = T(x' \to x) = 0 \qquad \text{if} \quad |x - x'| \geq c. \tag{4.2}$$

In this case, the detailed balance condition is trivially satisfied as

$$P(x) \cdot T(x \to x') = P(x') \cdot T(x' \to x) = 0. \tag{4.3}$$

- If $|x - x'| < c$, $\Delta x = x' - x$ and $-\Delta x = x - x'$ appear with the same probability $\frac{1}{2c}$. (More precisely, this is the probability density. The probability that $x' - x <$

$\Delta x < x' - x + \epsilon$ is $\epsilon/2c$.) By multiplying this and the probability of passing the Metropolis test, we obtain

$$T(x \to x') = \frac{1}{2c} \times \min(1, e^{S(x)-S(x')}), \tag{4.4}$$

$$T(x' \to x) = \frac{1}{2c} \times \min(1, e^{S(x')-S(x)}). \tag{4.5}$$

Suppose $S(x) \geq S(x')$. Then $e^{S(x)-S(x')} \geq 1$, and hence, a proposal $x \to x'$ passes the Metropolis test with 100% probability, and hence, $T(x \to x') = \frac{1}{2c}$. Therefore,

$$P(x) \cdot T(x \to x') = \frac{e^{-S(x)}}{Z} \times \frac{1}{2c}. \tag{4.6}$$

On the other hand, $e^{S(x')-S(x)} \leq 1$, and hence $T(x' \to x) = \frac{e^{S(x')-S(x)}}{2c}$. Therefore,

$$P(x') \cdot T(x' \to x) = \frac{e^{-S(x')}}{Z} \cdot \frac{e^{S(x')-S(x)}}{2c} = \frac{e^{-S(x)}}{Z} \times \frac{1}{2c}. \tag{4.7}$$

In this way, we could confirm that the detailed balance condition $P(x) \cdot T(x \to x') = P(x') \cdot T(x' \to x)$ is satisfied.
When $S(x) < S(x')$, we can easily check the detailed balance condition by repeating the same argument exchanging the roles of x and x'.

4.2 Calculation of Expectation Value

Let us see a concrete example of the calculation based on the Metropolis algorithm. As before, we use $S(x) = \frac{x^2}{2}$. Here is a sample code in C:

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

int main(void){
  int niter=100;  //Collect 100 samples.
  double step_size=0.5e0;  //Set step size to be 0.5.

  srand((unsigned)time(NULL));
  //Set the seeds of random numbers by using the system clock.

  /*********************************/
  /* Set the initial configuration */
  /*********************************/
  double x=0e0;
  int naccept=0;  //Counter for the number of acceptance.
  /*************/
  /* Main loop */
  /*************/
  for(int iter=1;iter<niter+1;iter++){
    double backup_x=x;
    double action_init=0.5e0*x*x;

    double dx = (double)rand()/RAND_MAX;
    dx=(dx-0.5e0)*step_size*2e0;
    x=x+dx;

    double action_fin=0.5e0*x*x;
    /*******************/
    /* Metropolis test */
    /*******************/
    double metropolis = (double)rand()/RAND_MAX;
    if(exp(action_init-action_fin) > metropolis)
      /* accept */
      naccept=naccept+1;
    else
      /* reject */
      x=backup_x;
    /***************/
    /* data output */
    /***************/
    printf("%.10f   %f\n",x,(double)naccept/iter);}
}
```

Let us decipher this sample code line by line. Firstly, we set the seed of the random number generator:

```
srand((unsigned)time(NULL));
```

Here, we are using the default random number generator in the system. It is not a good habit to use the same sequence of random numbers every time, so we used the system time as a seed. For more serious, large-scale simulations, we recommend you use a more sophisticated random number generator such as the Mersenne Twister [3].

Next, we set the initial condition. Here we chose $x = 0$:

```
double x=0e0;
int naccept=0;
```

`naccept` is a counter for the number of acceptances (i.e., how many times proposals of the update $x \to x'$ are accepted).

The following "main loop" is the main part. The variable `iter` corresponds to k. (`iter` means iteration.) An integer `niter` is the number of iterations, or equivalently, the number of configurations generated during the simulation. Before the Metropolis test, we do not know whether the proposed value will be accepted or not, so we save the value of `x` $= x^{(k)}$ in `backup_x`,

```
double backup_x=x;
```

then we calculate the action `action_init` $= S(x^{(k)})$ as

```
double action_init=0.5e0*x*x;
```

Here "init" means <u>init</u>ial, namely, this is the 'initial' action before the candidate x' is proposed. Next, `dx` $= \Delta x$ is randomly generated and $x' = x^{(k)} + \Delta x$ is calculated:

```
double dx = (double)rand()/RAND_MAX;
dx=(dx-0.5e0)*step_size*2e0;
x=x+dx;
```

Note that a random number between 0 and 1 is generated as `rand()/RAND_MAX`, then it is shifted and rescaled such that a uniform random number between $-c$ and $+c$ is obtained. By using x' obtained in this way, `action_fin` $= S(x')$ is calculated ("fin" means <u>fin</u>al). Finally, the Metropolis test is performed:

```
/*******************/
/* Metropolis test */
/*******************/
double metropolis = (double)rand()/RAND_MAX;
if(exp(action_init-action_fin) > metropolis)
  /* accept */
  naccept=naccept+1;
else
  /* reject */
  x=backup_x;
```

`metropolis` is a uniform random number between 0 and 1, which corresponds to r. Depending on the outcome of the Metropolis test, the candidate x' is accepted or rejected.

All programs based on Markov Chain Monte Carlo have essentially the same structure. Depending on the details of the problems, more sophisticated algorithms may be used, but essentially, any algorithm amounts to the improvement of $x \to x' = x + \Delta x$. Therefore, if you could understand how this program works, you can understand any complicated programs for MCMC except for technical details.

Let us see a result of an actual simulation. We take the initial configuration to be $x^{(0)} = 0$, and use the step size $c = 0.5$. (As we will see later, this choice of step size c is not optimal.) In Fig. 4.1, the distribution of $x^{(1)}, x^{(2)}, \ldots, x^{(K)}$ is shown for $K = 10^3, 10^5, 10^7$. We can clearly see the convergence to the target distribution $P(x) = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$ as K becomes larger. In Fig. 4.2, the expectation values $\langle x \rangle = \frac{1}{K}\sum_{k=1}^{K} x^{(k)}$ and $\langle x^2 \rangle = \frac{1}{K}\sum_{k=1}^{K}\left(x^{(k)}\right)^2$ are plotted. As K becomes larger, they converge to the correct values (i.e., the expectation values under the target distribution) 0 and 1.

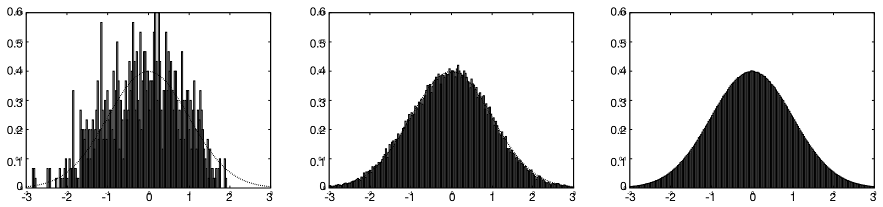

Fig. 4.1 The histogram of $x^{(1)}, x^{(2)}, \ldots, x^{(K)}$ for $K = 10^3$, 10^5 and 10^7. It converges to the target distribution $P(x) = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$ as K becomes large

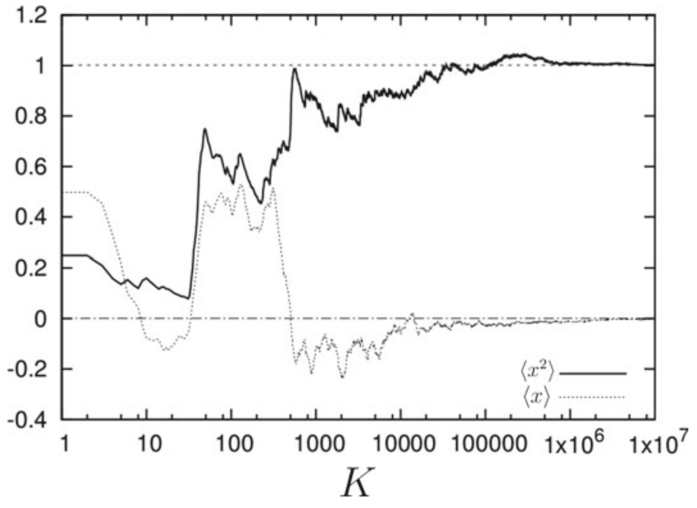

Fig. 4.2 $\langle x \rangle = \frac{1}{K}\sum_{k=1}^{K} x^{(k)}$ and $\langle x^2 \rangle = \frac{1}{K}\sum_{k=1}^{K} \left(x^{(k)}\right)^2$. They approach the correct values 0 and 1 as K increases

The step size c is taken such that the acceptance rate (the probability that the proposal $x \to x' = x + \Delta x$ is accepted) is not too large and not too small. If c is too large, the acceptance rate is very small, and the value of x rarely changes. If c is too small, the acceptance rate becomes almost 100%, but the change at each step is too small and hence x stays more or less the same value for a long time. Either way, compared to the optimal value of c, a lot more steps are needed in order to approximate the correct statistical distribution. This point is explained in detail in Sect. 4.3.

Typically, 30%–80% acceptance rate is the sweet spot. However, it depends on the algorithm and/or the kind of integral under consideration.

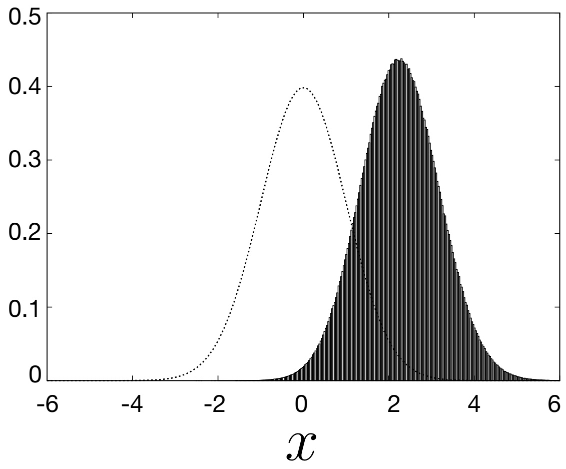

Fig. 4.3 The histogram of $K = 10^7$ configurations obtained by using the Metropolis algorithm, with a *wrong* choice $\Delta x \in \left[-\frac{1}{2}, 1\right]$. The dashed line is the target distribution $P(x) = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$. Because the detailed balance condition is not satisfied, the target distribution is not correctly reproduced

Example of Incorrect Implementation

People learn from their mistakes. To use the Metropolis algorithm correctly, let us see an example of a wrong way of using the algorithm. Suppose we chose Δx randomly between $-\frac{1}{2}$ and 1. Then the detailed balance condition is not satisfied; it can easily be seen by noticing that a transition $0 \to 1$ can happen with a nonzero probability while $1 \to 0$ cannot. The distribution of x calculated this way is shown in Fig. 4.3. Obviously, the target distribution is not correctly reproduced.

4.3 Autocorrelation

In principle, just by following the rules we have seen so far, we can always generate the target distribution. However, that it is correct in principle does not mean it is practically useful. Because we cannot live forever, we need to reach the target distribution as quickly as possible. Furthermore, we have to make sure that the samples we got are "good" ones. For example, even if we got 10,000 samples, if the acceptance rate is too low and we got only 10 different values, each of them appearing 1000 times, then the value of such "bad" samples is just the same as the value of 10 "good" samples.

The correlation is an important keyword to understand this issue. In Markov Chain Monte Carlo, because $x^{(k+1)}$ is obtained by slightly changing $x^{(k)}$, they are correlated to each other. This correlation is called the autocorrelation. If the step size is too small or the acceptance rate is too low, the autocorrelation can be very large. If the autocorrelation is larger, it takes a longer time for the convergence to the target probability distribution, and the quality of the samples becomes poorer. In this section, we will see how the autocorrelation can be estimated, and how it can be reduced.

4.3.1 Correlation with the Initial Value and Thermalization (Burn-in)

When we studied the Gaussian distribution $P(x) = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$, we took the initial configuration to be $x^{(0)} = 0$. We chose this value because we knew the center of the target distribution is $x = 0$. What could happen if we took the initial configuration far away from the center of the target distribution, say $x^{(0)} = 100$?

The result of the simulation with $x^{(0)} = 100$ is shown in Fig. 4.4. We can see that the value of x stays large for a while, because of a strong correlation with the initial value $x^{(0)} = 100$. After some time, the correlation with the initial configuration disappears, and the fluctuation about the center of the target distribution ($x = 0$) sets in. That the simulation reached the center of the distribution in this way is sometimes expressed as "the simulation (or the Markov chain) thermalized" or "the simulation reached thermalization". It is also said that "the Markov chain has burned in".

As the word "thermalization" suggests, intuition from physics is useful to understand this phenomenon. Imagine we put a small piece of ice into a cup of water. Until the ice completely melts, heat moves from water to ice. This is not a thermalized state, rather the effect from a particular choice of the initial condition ("put a small piece of ice into the water") is still there. However, after some time, the ice completely melts, temperature becomes uniform everywhere in the cup, and macroscopically we will not see a change anymore. This is the thermalized state. Each molecule is moving fast even in the thermalized state, but macroscopically it is just a "typical state". The analogous situation in the Markov Chain Monte Carlo simulation is that the corre-

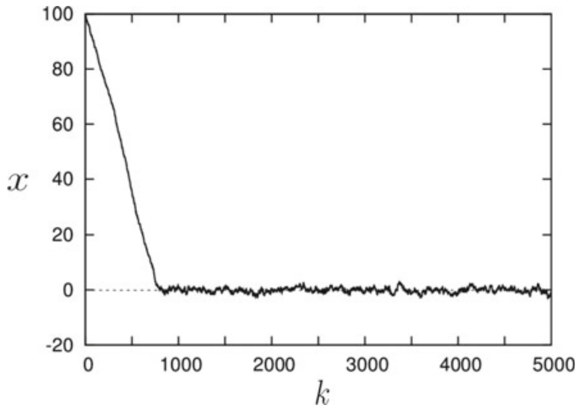

Fig. 4.4 The history of the simulation of the Gaussian distribution via the Metropolis algorithm. The step size is $c = 0.5$. Because we chose the initial value to be $x = 100$, which is very far from the typical values, the strong correlation with the initial value remains for a while, and it took a lot of steps to reach the typical values $|x| \sim 1$

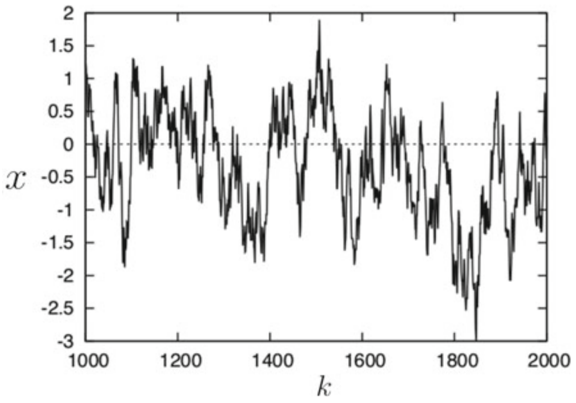

Fig. 4.5 The zoom-in of Fig. 4.4, from $k = 1000$ to $k = 2000$. Strong correlations survive at least for 20 or 30 steps

lation with the initial configuration became sufficiently small and the configurations are moving in the important regions dominating the integral. From this analogy, the meaning of the word "thermalization" should be clear. (As we will explain shortly, "thermalization" is used for another meaning as well; be cautious!)

When we calculate the expectation value, (unless the number of samples is extremely large) a large error arises if we use the non-thermalized configurations, because of the strong influence from the initial condition. Practically, we have to discard the samples before the thermalization. In Fig. 4.4, the thermalization is achieved by $k \sim 800$, so if we discard $k \leq 1000$ it should be more than enough. To make a more quantitative estimate, we should plot the expectation value calculated by using $k \geq K_{\text{cut}}$ as a function of K_{cut}. At sufficiently large K_{cut}, the dependence on K_{cut} disappears, which means that the effect of the initial condition disappeared.

When we study more complicated probability distributions, often we do not even know the rough shape of the target distribution. In such cases, we should plot several quantities, e.g., the energy or pressure if we are solving a physics problem. If they change monotonically, it is likely that the simulation has not thermalized yet. If the simulation reached thermalization, they would oscillate around the expectation values. In Fig. 4.5, at first x monotonically decreases, then it fluctuates about zero.

4.3.2 Autocorrelation

Some care is needed even after the thermalization. In Fig. 4.5, we zoomed in on the interval $1000 \leq k < 2000$ of Fig. 4.4. We can see the correlation between configurations, at least for 20 or 30 steps. This length (number of steps) that is needed for the autocorrelation to disappear is called the autocorrelation length. (A more quantitative estimate is given in Sect. 4.3.3.) If we treat correlated configurations as if they are independent, the statistical error is under-estimated. We cannot call two configurations independent unless they are separated by at least the autocorrelation length.

In order to estimate the expectation values precisely, sufficiently many independent configurations are needed. Otherwise, the expectation values fluctuate as the number of samples grows. The expression "the simulation thermalized" is used also to mean that sufficiently many independent configurations are collected, and the expectation values cease to fluctuate too much.

4.3.3 Jackknife Method

The Jackknife method [4–6] is an easy and effective way of estimating the autocorrelation. Here, for simplicity, we assume that the quantities we want to calculate can be obtained at each configuration.[2] Namely, we consider the quantities that can be expressed as a function of x as $f(x)$. For more generic cases, see Appendix D.

First we divide the samples into groups consisting of w configurations; the first group is $\{x^{(1)}, x^{(2)},\ldots,x^{(w)}\}$, the second group is $\{x^{(w+1)},x^{(w+2)},\ldots,x^{(2w)}\}$, and so on. Let the number of groups obtained this way be n. The average of $f(x)$ in the l-th group is

$$\tilde{f}^{(l,w)} \equiv \frac{1}{w} \sum_{j=(l-1)w+1}^{lw} f(x^{(j)}). \tag{4.8}$$

By using $\tilde{f}^{(l,w)}$, the Jackknife error is defined as

$$\Delta_w \equiv \sqrt{\frac{1}{n(n-1)} \sum_{l=1}^{n} \left(\tilde{f}^{(l,w)} - \bar{f}\right)^2}. \tag{4.9}$$

Here $\bar{f}$ is the average of $f(x)$ obtained by using all configurations. Namely, the Jackknife error is the standard error obtained by treating each of the n groups as an independent sample and regarding $\tilde{f}^{(l,w)}$ as the value obtained from those independent samples.

If there are sufficiently many samples, Δ_w grows gradually with w, and beyond some point (say at $w \geq w_c$) Δ_w becomes almost constant. The values of w_c and Δ_{w_c} obtained in this way give a reasonable estimate of the autocorrelation length and the statistical error, respectively.

In Fig. 4.6, the expectation value of x^2 and the Jackknife error Δ_w are shown. The error bar spreads quickly up to $w = 20$ or so, but after $w = 40$ there is almost no change. Hence $w = 50$ should be a safe choice. In Fig. 4.7, the values of $\tilde{f}^{(l,w)}$ calculated with $w = 50$ are plotted. Almost no autocorrelation is visible, and hence, we can safely regard each group as an independent sample. The result obtained this

[2] Quantities not in this class include the variance of the probability distribution. Another example is the mass of a composite particle in a physics simulation.

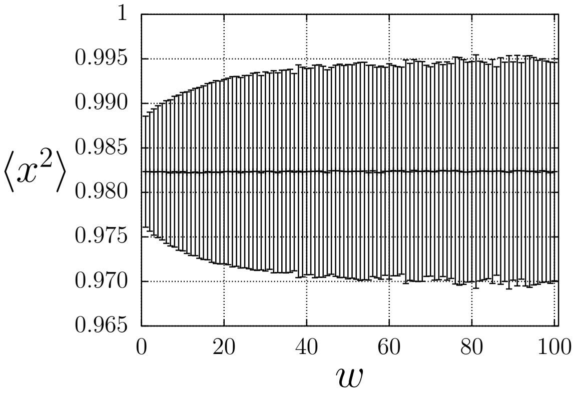

Fig. 4.6 The expectation value of x^2 and the Jackknife error

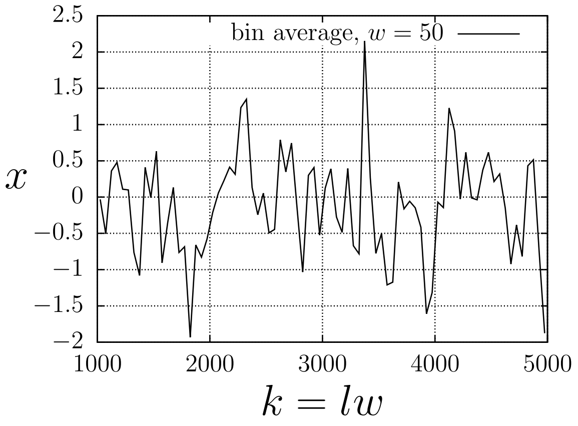

Fig. 4.7 The average of each group $\tilde{f}^{(l,w)}$, with the width $w = 50$

way was $\langle x^2 \rangle = 0.982 \pm 0.012$, which is in a reasonably good agreement with the analytic value $\langle x^2 \rangle = 1$.

Let us calculate a little bit by hand, to understand why the autocorrelation length can be estimated by using the Jackknife method. We consider two different widths w and $2w$ for the grouping. Then, by construction,

$$\tilde{f}^{(l,2w)} = \frac{\tilde{f}^{(2l-1,w)} + \tilde{f}^{(2l,w)}}{2}. \tag{4.10}$$

If n groups are obtained when the width is w, then $\frac{n}{2}$ groups are obtained when the width is $2w$. Hence the Jackknife error obtained by using the width $2w$ is

$$\Delta_{2w} = \sqrt{\frac{1}{\frac{n}{2}\left(\frac{n}{2}-1\right)}\sum_{l=1}^{n/2}\left(\tilde{f}^{(l,2w)}-\overline{f}\right)^2}$$
$$= \sqrt{\frac{4}{n(n-2)}\sum_{l=1}^{n/2}\left(\frac{\left(\tilde{f}^{(2l-1,w)}-\overline{f}\right)}{2}+\frac{\left(\tilde{f}^{(2l,w)}-\overline{f}\right)}{2}\right)^2}. \tag{4.11}$$

If the w is sufficiently large such that $\tilde{f}^{(2l-1,w)}-\overline{f}$ and $\tilde{f}^{(2l,w)}-\overline{f}$ can be interpreted as independent samples fluctuating about zero, then the products of independent quantities, $\left(\tilde{f}^{(2l-1,w)}-\overline{f}\right)\cdot\left(\tilde{f}^{(2l,w)}-\overline{f}\right)$, should be averaged to zero when summed over l. Therefore, approximately,

$$\Delta_{2w} \sim \sqrt{\frac{1}{n^2}\sum_{l=1}^{n}\left(\tilde{f}^{(l,w)}-\overline{f}\right)^2} \sim \Delta_w \tag{4.12}$$

has to hold. (Note that we assumed that n is sufficiently large.) Therefore, when w is larger than the autocorrelation length, Δ_w is almost constant.

4.3.4 Adjustment of the Step Size

To perform the simulation efficiently, we have to tune the parameters appropriately such that *more independent samples are generated with less cost.*[3] In the current case, the step size is the parameter to be tuned.

In the Metropolis algorithm, the candidate of the new configuration is accepted with the probability $\min(1, e^{-\Delta S})$, where ΔS is the increment of the action S. Let us consider the case of the Gaussian integral $S(x) = \frac{x^2}{2}$ again, and suppose that the transition from $x \sim 0$ to $x + \Delta x$ was proposed. Then, if $\Delta x \gg 1$, the acceptance probability is $\min(1, e^{-\Delta S}) = e^{-\Delta S} \ll 1$, namely, the candidate is rejected almost with 100% probability. In other words, only $\Delta x \lesssim 1$ has a reasonable chance of being accepted. If the step size c is too large, $\Delta x \lesssim 1$ is obtained only with probability $\frac{1}{c}$. Hence, the acceptance rate becomes very small. Furthermore, even when the value of x is updated, it just means the change Δx was of order 1, regardless of the value of c. Therefore, the acceptance rate is sacrificed for nothing, and the autocorrelation length increases proportionally to c. In such a parameter region, the product of the step size c and the acceptance rate becomes constant. On the other hand, if c becomes too small, the acceptance rate is almost 100%, but the amount of the change at each step decreases proportionally to c. Such a process can be regarded as the random

[3] Here, we assume that the "cost" is the amount of computation, or equivalently the time or the electricity bills for the computation. When we use parallel computers, we can save time by paying more electricity bills, so the notion of "cost" is not trivial.

Table 4.1 The relation between the step size c and the acceptance rate, measured from 10,000 samples

Step size c	Acceptance rate	$c \times$ acceptance rate
0.5	0.9077	0.454
1.0	0.8098	0.810
2.0	0.6281	1.256
3.0	0.4864	1.459
4.0	0.3911	1.564
6.0	0.2643	1.586
8.0	0.1993	1.594

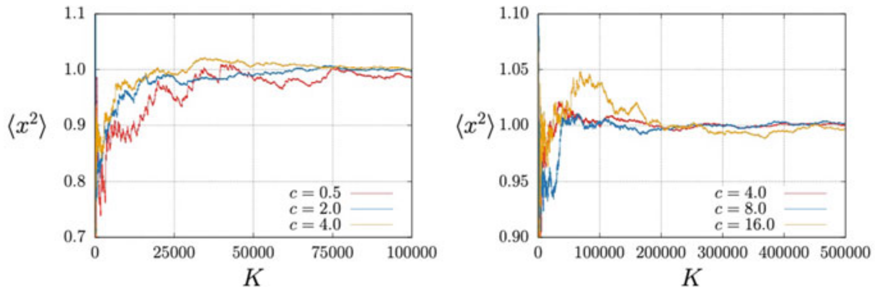

Fig. 4.8 $\langle x^2 \rangle = \frac{1}{K}\sum_{k=1}^{K}\left(x^{(k)}\right)^2$ with several different choices of the step size c. The simulation is not efficient when the step size is too big or too small

walk with the step size c. Typically, in the random walk, the change of the value of x after n steps is $c\sqrt{n}$. For this reason, the autocorrelation length is proportional to $\frac{1}{c^2}$. Therefore, the autocorrelation length will be minimized when c is not too large and not too small.

In Table 4.1, the acceptance rates at several values of c are summarized. The product of c and the acceptance rate is almost constant at $c > 4$, which suggests that c is too large there. At $c = 0.5$ and $c = 1.0$ the acceptance rate is high, which suggests c is too small. Hence, $c = 2.0 \sim 4.0$ appears to be the optimal choice.

In Fig. 4.8, we showed how $\langle x^2 \rangle$ converges to 1, for several values of c. With $c = 2.0$ or $c = 4.0$, faster convergence can be seen compared to smaller or larger c.

4.3.5 Box-Muller Method Revisited

The Box-Muller method introduced in Sect. 2.4.1 can be regarded as a special kind of Markov Chain Monte Carlo. Let us call the Gaussian random numbers generated by the Box-Muller method as $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$. We can confirm that they satisfy the conditions for MCMC:

- It is a Markov chain, namely, the probability that $x^{(k+1)}$ is obtained does not depend on $x^{(0)}, x^{(1)}, \ldots, x^{(k-1)}$. This condition is satisfied trivially—in fact $x^{(k+1)}$ does not even depend on $x^{(k)}$. Expressed as an equation, the transition probability is $T(x \to x') = P(x')$.
- The irreducibility is also trivially satisfied. Any transition can happen just by one step.
- It is easy to confirm the aperiodicity as well. Starting from x, there is a chance to come back to x after any number of steps.
- The detailed balance condition can be confirmed as follows:

$$P(x) \cdot T(x \to x') = P(x') \cdot T(x' \to x) = P(x) \cdot P(x'). \tag{4.13}$$

In the usual Markov Chain Monte Carlo methods such as the Metropolis algorithm, $x^{(k+1)}$ is obtained by slightly changing $x^{(k)}$, which inevitably leads to autocorrelation. In the Box-Muller method, $x^{(k+1)}$ is created without referring to $x^{(k)}$ and hence there is no autocorrelation. This is the reason that it is called a "random number". In this sense, the Box-Muller method is much better than the Metropolis algorithm. On the other hand, such efficient algorithms are known only for simple probability distributions. The Metropolis algorithm is very powerful because it can be applied to any probability distribution.

The Metropolis-Hastings algorithm (Sect. 5.3) and the Gibbs sampling algorithm (Sect. 5.2) are based on a very simple idea: less autocorrelation leads to higher efficiency. The Box-Muller method can also be regarded as a special case of these algorithms.

4.4 Examples Other Than the Gaussian Distribution

So far we have studied only the Gaussian distribution $P(x) = \frac{e^{-S(x)}}{Z} = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$. Some readers may wonder if such a simple method can always work, so let us see a few other examples.

First, we consider the superposition of two Gaussian distributions,

$$P(x) = \frac{e^{-\frac{(x-3)^2}{2}} + e^{-\frac{(x+3)^2}{2}}}{2\sqrt{2\pi}}. \tag{4.14}$$

The action $S(x)$ can be taken as

$$S(x) = -\log\left(e^{-\frac{(x-3)^2}{2}} + e^{-\frac{(x+3)^2}{2}}\right). \tag{4.15}$$

Hence we only have to rewrite two lines in the sample code, namely, we change

```
action_init=0.5e0*x*x;
```

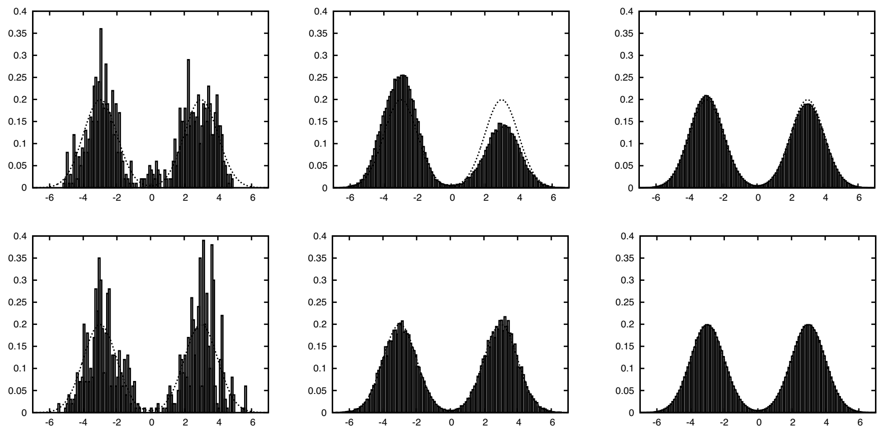

Fig. 4.9 The histograms of $x^{(1)}, x^{(2)}, \ldots, x^{(K)}$ with $K = 10^3, 10^5, 10^7$. The dashed lines are the target distribution $P(x) = \frac{e^{-\frac{(x-3)^2}{2}} + e^{-\frac{(x+3)^2}{2}}}{2\sqrt{2\pi}}$. The step size is 0.5 (top) or 5.0 (bottom). When the step size is 0.5, the deviation from the target distribution is visible even with 10^7 configurations. When the step size is 5.0, the convergence to the target distribution is much faster

and

```
    action_fin=0.5e0*x*x;
```

to

```
action_init=-log(exp(-0.5e0*(x-3e0)*(x-3e0))+exp(-0.5e0*
(x+3e0)*(x+3e0)));
```

and

```
action_fin=-log(exp(-0.5e0*(x-3e0)*(x-3e0))+exp(-0.5e0*
(x+3e0)*(x+3e0)));
```

We performed simulations with step sizes 0.5 and 5.0, and have shown the histograms of x in Fig. 4.9. The top and bottom rows are for step size 0.5 and 5.0, respectively. The target distribution $P(x) = \frac{e^{-\frac{(x-3)^2}{2}} + e^{-\frac{(x+3)^2}{2}}}{2\sqrt{2\pi}}$ is shown with the dashed lines. We can see the convergence to the target distribution for both cases. When the step size is 5.0, convergence is achieved quickly. However, when the step size is 0.5, convergence is slower; even with 10^7 samples, the heights of the two peaks do not completely agree. What is the reason?

In the importance sampling, the configurations with smaller weights are avoided. In the current setup with two peaks, a bottleneck near $x = 0$ consists of low-weight configurations that are not sampled frequently. Therefore, if the step size is small, it is difficult to go through this bottleneck and reach the other peak. If the transitions between the two peaks happen frequently, the heights quickly become the same. Otherwise, more time is spent at one of the peaks and the heights differ for a long time.

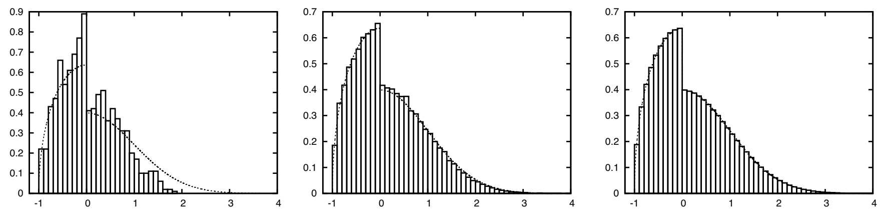

Fig. 4.10 The probability distribution (4.16) reproduced by using the Metropolis algorithm with the step size 0.5. The histograms of $x^{(1)}, x^{(2)}, \ldots, x^{(K)}$ with $K = 10^3, 10^5, 10^7$ are shown

As explained in Sect. 3.2, such a bottleneck can effectively break the irreducibility. (Of course, the irreducibility is not completely broken, so if we wait very long we will get the right answer. However a "very long time" can often be more than the human lifespan.) When the step size is 5.0, configurations can jump over the bottleneck and go directly from one peak to the other, so a quick convergence can be achieved. In Sect. 4.7.1, we consider a more extreme example, $P(x) = \frac{e^{-\frac{x^2}{2}} + e^{-\frac{(x-100)^2}{2}}}{2\sqrt{2\pi}}$. Please think about the optimum step size for that case. (In Sect. 6.3.3, we will introduce the replica-exchange method, which can be used for more complicated problems.)

As yet another example, let us consider a combination of the semi-circle distribution at $x < 0$ and the Gaussian distribution at $x \geq 0$:

$$P(x) = \begin{cases} \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} & (x \geq 0) \\ \frac{2}{\pi}\sqrt{1 - x^2} & (-1 \leq x < 0) \\ 0 & (x < -1) \end{cases} \tag{4.16}$$

The probability is zero at $x < -1$, hence $x' < -1$ is always rejected. (Equivalently, we take $S(x) = \infty$ at $x < -1$.) The distribution obtained in this way is shown in Fig. 4.10. As expected, $x < 0$ is a semi-circle, and $x > 0$ is Gaussian. In this case, the choice of the step size is not as important as in the last example because there is no bottleneck.

4.5 Application to Complicated Integrals

In Markov Chain Monte Carlo, the expectation values can be obtained, but the partition function Z cannot be calculated directly. In many cases, the partition function is merely a normalization factor that is not particularly useful. However, sometimes we happen to be interested in the value of Z itself. In such a case, how can we calculate it?

When there is only one variable, we can plot the probability distribution $P(x) = \frac{e^{-S(x)}}{Z}$ obtained via MCMC and take the ratio with $e^{-S(x)}$. However, this method does not work when there are many variables. Below, we show a method that can easily be generalized to multivariate distributions.

The action $S(x)$ can be any complicated real-valued function, as long as the partition function $Z = \int dx e^{-S(x)}$ is finite. If the action is a simple function, for example $S_0(x) = \frac{x^2}{2}$, the partition function can be calculated analytically, as $Z_0 = \int dx e^{-S_0(x)} = \sqrt{2\pi}$. By using such an S_0, we can calculate the ratio of Z and Z_0 via MCMC:

$$\frac{Z}{Z_0} = \frac{1}{Z_0}\int dx e^{-S_0} \cdot e^{S_0 - S} = \left\langle e^{S_0 - S}\right\rangle_0 . \tag{4.17}$$

Here $\langle \cdot \rangle_0$ is the expectation value calculated by using e^{-S_0} as the weight. If we know Z_0, we can get the value of Z as well.

As an example, let us consider

$$S(x) = \begin{cases} -\frac{1}{2}\log(1-x^2) & (-1 < x < 1) \\ \infty & (x < -1, x > 1) \end{cases} \tag{4.18}$$

Then the partition function Z has to be the area of the semi-circle, $\frac{\pi}{2}$. If we calculate the expectation value of $e^{S_0(x)-S(x)}$ which is expressed as

$$e^{S_0(x)-S(x)} = \begin{cases} e^{\frac{1}{2}x^2}\sqrt{1-x^2} & (-1 < x < 1) \\ 0 & (x < -1, x > 1) \end{cases} \tag{4.19}$$

we should obtain

$$\frac{Z}{Z_0} = \frac{\pi}{2} \cdot \frac{1}{\sqrt{2\pi}} = 0.6266.... \tag{4.20}$$

In the left panel of Fig. 4.11, we can see the convergence to this value.

In principle, this method always works. In practice, however, it works only when the probability distributions $P(x) = \frac{e^{-S(x)}}{Z}$ and $P_0(x) = \frac{e^{-S_0(x)}}{Z_0}$ have a sufficiently large overlap. To illuminate this point, let us consider $S(x) = \frac{(x-\alpha)^2}{2}$. In this case, $P(x)$ and $P_0(x)$ are peaked around $x = \alpha$ and $x = 0$, respectively. The weight factor to be calculated via MCMC is

$$\frac{P(x)}{P_0(x)} = e^{S_0(x)-S(x)} = e^{\frac{x^2}{2} - \frac{(x-\alpha)^2}{2}} . \tag{4.21}$$

If α is very large, say $\alpha = 100$, the values of $e^{S_0 - S}$ appearing in the simulation are almost always extremely small, e.g., e^{-5000}, because the value of x in the probability distribution P_0 is typically 1. However, once in e^{+5000} configurations or so, x can become as large as 100, and then $e^{S_0 - S}$ takes a large value like e^{+5000}. After taking the average over infinitely many configurations, we obtain $\frac{Z}{Z_0} = 1$. However, we will not see the configurations dominating this average, $x \sim 100$, during a realistic

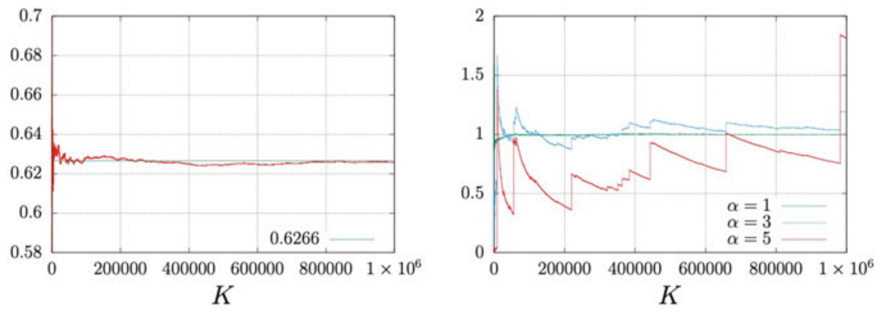

Fig. 4.11 [Left] The expectation value of (4.19) calculated by using the Metropolis algorithm with step size 4. [Right] The expectation value of (4.21) calculated via the Metropolis algorithm with step size 4. The parameter α is $1, 3$, or 5. The horizontal axis is the number of configurations K used for the calculation of the expectation value

simulation time. Obviously, we cannot get the right answer in any practical sense. In the right panel of Fig. 4.11, we can see that the convergence is getting significantly slow already at $\alpha = 3$. At $\alpha = 5$, we see a large deviation from the right answer $\langle e^{\frac{x^2}{2} - \frac{(x-\alpha)^2}{2}} \rangle_0 = 1$ even with 1,000,000 configurations. This issue is called the overlap problem because it happens due to the lack of overlap between $P(x)$ and $P_0(x)$.[4]

In this example, the overlap problem can be resolved rather easily. Let us be less ambitious and split the problem into M tasks. We choose a chain of actions S_0, S_1, S_2, ..., $S_M = S$, in such a way that S_n and S_{n+1} are sufficiently close. For example, we can use $S_n = \frac{1}{2}\left(x - \frac{\alpha n}{M}\right)^2$, with $\frac{\alpha}{M} \sim 1$. Then $\frac{Z_{n+1}}{Z_n}$ (where $Z_n \equiv \int dx e^{-S_n(x)}$) can be calculated without suffering from a serious overlap problem. By calculating $\frac{Z_1}{Z_0}, \frac{Z_2}{Z_1}, \ldots, \frac{Z_M}{Z_{M-1}}$, we can determine $Z = Z_M$. Similar methods can be used for more complicated $S(x)$ and Z as well. One of the authors applied this method to a complicated integral in a physics problem [7].

4.6 Sign Problem

All the arguments above assumed $e^{-S(x)} \geq 0$ so that it can be regarded as a probability. This assumption is not valid in many important applications in physics, i.e., the weight $e^{-S(x)}$ can be negative or complex at certain values of x. Then the Markov Chain Monte Carlo methods are not directly applicable. This is the infamous sign problem.[5] For reviews, see e.g., Refs. [8, 9]. The sign problem in physics is one of the biggest motivations for the quantum computer [10, 11].

[4] The negative sign problem, which will be discussed in Sect. 4.6, can be regarded as a version of the overlap problem.

[5] When $e^{-S(x)}$ is complex, it is also called the "phase problem", but even in that case "sign problem" is more commonly used.

There is no generic solution to the sign problem known to date. The sign problem is a very difficult problem that is related to one of the biggest issues in computer science and mathematics: the $P \neq NP$ conjecture. If there were a generic solution, it would mean $P = NP$ [12], and hence it is widely believed that a generic solution cannot exist. However, several case-by-case solutions specific to concrete problems are known. Also, sometimes we can beat the sign problem by brute force, simply by investing a lot of computational resources. In this section, we introduce a typical example of such a brute-force method: the reweighting method.

Suppose $e^{-S(x)}$ is complex. We write it as a product of the absolute value $|e^{-S(x)}| = e^{-S_0(x)}$ and the complex phase $e^{i\theta(x)}$:

$$e^{-S(x)} = e^{-S_0(x)} \times e^{i\theta(x)}. \tag{4.22}$$

It is straightforward to perform MCMC by adopting e^{-S_0} as the weight. (This is called the phase-quenched simulation.) We use $\langle \cdot \rangle_0$ to denote the expectation value with this weight. Then

$$\int dx e^{-S(x)} = \langle e^{i\theta(x)} \rangle_0 \times \int dx e^{-S_0(x)}, \tag{4.23}$$

and hence, by calculating $\langle e^{i\theta(x)} \rangle_0$ and $\int dx e^{-S_0(x)}$ we can determine $\int dx e^{-S(x)}$. Note that we can use the method explained in Sect. 4.5 to calculate $\int dx e^{-S_0(x)}$.

This method is simple in principle, but often $e^{i\theta}$ fluctuates very violently and $\langle e^{i\theta} \rangle_0$ becomes very close to zero. This is often the case when there are many variables in the integral. In case such large fluctuations appear, we need to determine $\langle e^{i\theta} \rangle_0$ very precisely, which makes the simulation harder.

The calculation of the expectation value $\langle f(x) \rangle$ is also simple, by using

$$\langle f(x) \rangle = \frac{\langle f(x) e^{i\theta} \rangle_0}{\langle e^{i\theta} \rangle_0}. \tag{4.24}$$

In this case, again, both the numerator and denominator can become very small if $e^{i\theta}$ fluctuates a lot.

The sign problem becomes particularly severe when the presence of the phase factor affects the configurations dominating the integral.[6] In such cases, the sign problem can be regarded as a version of the overlap problem.

[6] In such cases, contributions from dominant configurations in the phase-quenched simulation are canceled by particularly violent oscillations of $e^{i\theta}$ and a new peak emerges from the tail of the phase-quenched probability distribution.

4.7 Common Mistakes

In this section, we show a few common mistakes among beginners. These mistakes are sometimes made by experts, too.

4.7.1 *Changing Step Size in the Middle of the Simulation*

Imagine there is a bottleneck in the probability distribution, like in Fig. 4.9. As an extreme example, we can consider $S(x) = -\log\left(e^{-\frac{x^2}{2}} + e^{-\frac{(x-100)^2}{2}}\right)$. Then $e^{-S(x)}$ has two peaks around $x = 0$ and $x = 100$, in between $e^{-S(x)}$ is almost zero. In such cases, it is not easy to sample the entire distribution because it is difficult to go through the bottleneck. Also, it often happens that the simulation is trapped in a special configuration surrounded by bottlenecks, gets stuck there for a long time, and the acceptance rate becomes very low. Then, you might be tempted to change the step size, say make it smaller so that the acceptance rate goes up. However, if you do that, you get a wrong result. *You must not change the step size in the middle of the simulation.*

Still, it is totally fine to combine multiple step sizes. *As long as the conditions explained in Chap.* 3 *are not broken, we can do whatever we like*. For example, you can take step size $c = 1$ for the even steps and $c = 100$ for the odd steps. With this choice, at the odd steps, the transition between the peaks around $x = 0$ and $x = 100$ can take place, and hence the entire distribution can be sampled (Fig. 4.12). Alternatively, we can set the step size by throwing a dice. Namely, at each step, you can choose the step size to be $c = 1, 2, 3, 4, 5$ or 6 with probability $\frac{1}{6}$ for each value, without breaking the conditions listed in Chap. 3. We can do whatever, so we recommend you try various options.

When the configurations are far from thermalization, the simulation often gets stuck at a special configuration. To avoid this, we can choose a good initial configuration (if we know about basic features of the probability distribution), or we can take the step size to be small at the beginning and then switch to a larger step size after thermalization. There is no problem as long as we use the same step size when we calculate the expectation values. Because the configurations before thermalization are simply discarded, we can change the step size during the thermalization process.[7]

[7] Another common technique is to skip the Metropolis test at the early stage of the thermalization process. This technique is particularly powerful when it is combined with the HMC algorithm introduced in Sect. 5.1. Because those configurations are not used for the calculation of the expectation value, there is no need for the detailed balance condition there.

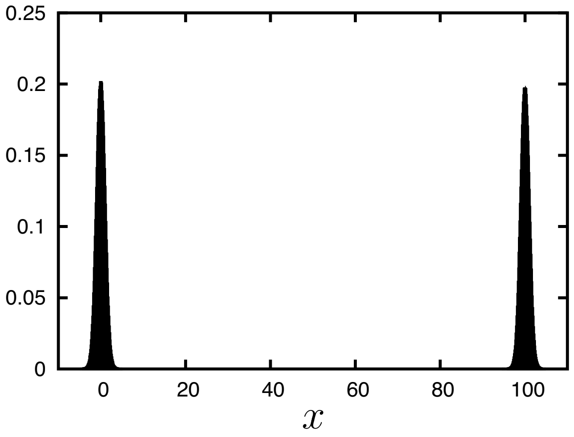

Fig. 4.12 The probability distribution $P(x) = \frac{e^{-\frac{x^2}{2}} + e^{-\frac{(x-100)^2}{2}}}{2\sqrt{2}}$ reproduced by using the Metropolis algorithm with the step size $c = 1$ for the even steps and $c = 100$ for the odd steps. The histogram of 10,000,000 samples is shown. The target distribution $P(x)$ is also drawn, but it is invisible to the naked eyes because it agrees well with the histogram

4.7.2 Mixing the Configurations Obtained by Using Different Step Sizes

This example is similar to the one in Sect. 4.7.1. If several sequences of the configurations are obtained by using different step sizes, the convergence to the right answer is guaranteed for each step size. However, if the simulations are terminated at finite numbers of steps and the configurations obtained by using different step sizes are mixed, the statistical error might become uncontrollable, and unreasonable results may be obtained. Still, if the autocorrelation length is properly estimated for each step size, it is possible to take the average by mixing the independent samples.

If several sequences of the configurations are obtained by using the same step size, there is no problem in using all the configurations for the analyses, as long as each run is sufficiently well thermalized.

4.7.3 "Random Numbers" Were Not Really Random

As we have mentioned, the "random numbers" used in numerical simulations are actually pseudorandom. Therefore, we can make unexpected mistakes if we are not careful enough. For example, what happens if we repeat the same sequence of pseudorandom numbers every 1000 steps? The outcome is shown in Fig. 4.13. Obviously, the answer is wrong.

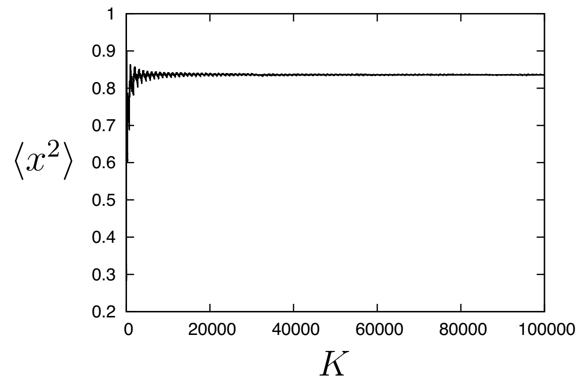

Fig. 4.13 *Wrong* example of the Gaussian integral with Metropolis algorithm, step size $c = 1$. The same sequence of pseudorandom numbers was repeated every 1000 steps (i.e., "random numbers" were not really random). In the correct simulation, $\langle x^2 \rangle = \frac{1}{K}\sum_{k=1}^{K}\left(x^{(k)}\right)^2$ has to converge to 1. However, in this example, convergence to a wrong value is observed

Such mistakes happen very easily. Suppose we work on a very large-scale simulation that takes several months. Then we have to split the task into small jobs which can be finished at a short time scale, say one day or an hour. For example, 10 steps are processed in each job, and we repeat 1000 jobs so that 10000 steps are obtained in total. Then if we make an error in the setting of the seeds, the same sequence of pseudorandom numbers would be repeated every 10 steps.

In order to avoid such mistakes, it is better to save both the configuration and the information regarding the pseudorandom numbers at the end of each job. It is kind of tedious at first, but once you write a code, you can copy-and-paste the same one.

4.8 Multivariate Metropolis Algorithm

So far, we have only dealt with the univariate distributions. Although all the essence is there, it is instructive to learn about the cases with many variables that suffer from the curse of dimensionality. In this section, we generalize the Metropolis algorithm to multivariate distributions.

This generalization is very simple. Let the variables be $(x_1, x_2, \ldots, x_n)$. Because there are multiple variables, there are roughly two possible ways we can update them. The first one is ⟨Update simultaneously⟩:

Multivariate Metropolis ⟨Update simultaneously⟩

1. For all $i = 1, 2, \ldots, n$, choose Δx_i randomly from $[-c_i, +c_i]$, and propose $x'_i \equiv x_i^{(k)} + \Delta x_i$ as a candidate of $x_i^{(k+1)}$. The step sizes $c_1, c_2, \ldots, c_n$ can be different from each other.
2. Metropolis test: the candidate $\{x'\}$ is accepted and the value of $\{x\}$ is updated as $\{x^{(k+1)}\} = \{x'\}$ with a probability $\min(1, e^{S(\{x^{(k)}\})-S(\{x'\})})$. Otherwise $\{x'\}$ is rejected and the value of $\{x\}$ remains unchanged, as $\{x^{(k+1)}\} = \{x^{(k)}\}$.

The other one is ⟨Update one by one⟩:

Multivariate Metropolis ⟨Update one by one⟩

1. Choose Δx_1 randomly from $[-c_1, +c_1]$, and take $x'_1 \equiv x_1^{(k)} + \Delta x_1$. Other variables are left untouched, $x'_i \equiv x_i^{(k)}$ $(i = 2, 3, \ldots, n)$.
2. Metropolis test: the candidate $\{x'\}$ is accepted and the value of $\{x\}$ is updated as $\{x^{(k+1)}\} = \{x'\}$ with a probability $\min(1, e^{S(\{x^{(k)}\})-S(\{x'\})})$. Otherwise $\{x'\}$ is rejected and the value of x remains unchanged, as $\{x^{(k+1)}\} = \{x^{(k)}\}$. (Whether the proposal is accepted or not, $x_2, x_3, \ldots, x_n$ are left untouched.)
3. Update x_2 in the same manner. ($x_1, x_3, \ldots, x_n$ are left untouched.)
4. In the same manner, update $x_3, \ldots, x_n$ one by one.

For the Metropolis test, a uniform random number r between 0 and 1 is generated and the candidate $\{x'\}$ is accepted if $r < e^{S(\{x^{(k)}\})-S(\{x'\})}$.

We recommend that the reader check that, either way, the four conditions of Markov Chain Monte Carlo explained in Chap. 3 are satisfied. (Strictly speaking, the situation is a little bit subtle in ⟨Update one by one⟩; see the exercise at the end of this chapter.)

When there are many variables, ⟨Update simultaneously⟩ usually forces us to take the step size c_i small, because otherwise the acceptance rate becomes low. On the contrary, ⟨Update one by one⟩ allows us to take the step size relatively large. Furthermore, in case the variable x_i interacts with only a few other variables (e.g., the nearest-neighbor interaction which can be written as $S = f(x_1, x_2) + f(x_2, x_3) + \ldots f(x_{n-1}, x_n)$) x_i can be updated by calculating only the terms containing x_i (in the example above, $f(x_{i-1}, x_i) + f(x_i, x_{i+1})$), and hence, the computational cost can be reduced.

Either way, a different value of step size c_i can be used for each variable x_i. Please check that the detailed balance condition is still satisfied, as an instructive exercise. In case the widths of the probability distribution heavily depend on the variables, (there can be exceptions, but in many cases) the simulation becomes more efficient if c_i is proportional to the width. When we do not have a good guess regarding the width, we can try several different step sizes and see how the acceptance rate changes.

4.8.1 Multivariate Gaussian Distribution

Let us consider the multivariate Gaussian distribution[8]

$$S(x_1,\dots,x_n)=\frac{1}{2}\sum_{i,j=1}^{n}A_{ij}x_ix_j \qquad (A_{ij}=A_{ji}). \tag{4.25}$$

As an example, we take $n=2$. We write $x_1=x$ and $x_2=y$, and choose the coefficients to be $A_{11}=1$, $A_{22}=1$, $A_{12}=\frac{1}{2}$. Then, the action $S(x,y)$ becomes[9]

$$S(x,y)=\frac{x^2+y^2+xy}{2}. \tag{4.26}$$

The term $\frac{1}{2}xy$ in $S(x,y)$ introduces the correlation between x and y. For example, we can imagine that x and y parametrize a person's mathematical skills and baseball skills, respectively. Larger x means better mathematical skills, and larger y means better baseball skills. Zero is average, and a large negative value means bad skill. If $S(x,y)=\frac{x^2+y^2}{2}$ and $P(x,y)\propto e^{-\frac{x^2+y^2}{2}}$, the values of x and y are not correlated at all, which means whether one is good or bad at mathematics is not related to baseball skills. If $S(x,y)=\frac{x^2+y^2+xy}{2}$ and $P(x,y)\propto e^{-\frac{x^2+y^2+xy}{2}}$, it is unlikely that one is good both at math and baseball or bad both at math and baseball, rather if one is good at math or baseball he/she is likely to be bad at the other. It would be a reasonable assumption because they have to split a finite amount of time to the study of mathematics and practice of baseball, and also because God usually does not bless a person twice.

We use the Metropolis algorithm to generate this probability distribution. If the number of variables is as small as 2, there is no big difference between ⟨Update simultaneously⟩ and ⟨Update one by one⟩. Here we use the former. Below we show some sample code written in C:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

int main(void){
  int niter=10000;
  double step_size_x=0.5e0;
  double step_size_y=0.5e0;
```

[8] See Appendix B.2 for basic properties.

[9] If we consider a slightly more generic version $S(x,y)=\frac{x^2+y^2+2Axy}{2}$, the same kind of calculation applies to $-1<A<1$. If $A\geq 1$ or $A\leq -1$, then $P(x,y)\propto e^{-S(x,y)}$ does not make sense as a probability distribution. Why? (Hint: See Sect. 6.1.2.)

```
  srand((unsigned)time(NULL));
  /*********************************/
  /* Set the initial configuration */
  /*********************************/
  double x=0e0;
  double y=0e0;
  int naccept=0;
  /*************/
  /* Main loop */
  /*************/
  for(int iter=1;iter<niter+1;iter++){
    double backup_x=x;
    double backup_y=y;
    double action_init=0.5e0*(x*x+y*y+x*y);

    double dx = (double)rand()/RAND_MAX;
    double dy = (double)rand()/RAND_MAX;
    dx=(dx-0.5e0)*step_size_x*2e0;
    dy=(dy-0.5e0)*step_size_y*2e0;
    x=x+dx;
    y=y+dy;
    double action_fin=0.5e0*(x*x+y*y+x*y);
    /*******************/
    /* Metropolis test */
    /*******************/
    double metropolis = (double)rand()/RAND_MAX;
    if(exp(action_init-action_fin) > metropolis){
      /* accept */
      naccept=naccept+1;
    }else{
      /* reject */
      x=backup_x;
      y=backup_y;}
    /***************/
    /* data output */
    /***************/
    // output the results every ten steps.
    if(iter%10==0){
      printf("%.10f   %.10f   %f\n",x,y,(double)naccept/
      iter);}
  }
}
```

The code is almost identical to the one for $S(x) = \frac{x^2}{2}$ shown in Sect. 4.2. The only differences are that there is one more variable y and $S(x) = \frac{x^2}{2}$ is replaced with $S(x, y) = \frac{x^2+y^2+xy}{2}$. The same applies even if there are thousands or millions of variables, and even if the probability distribution is a very complicated function.

As we have seen, different step sizes can be used for x and y. However, in this specific example, it is natural to use the same step size because x and y appear in $S(x, y)$ symmetrically. We used the Metropolis algorithm with the step size $c = 0.5$,

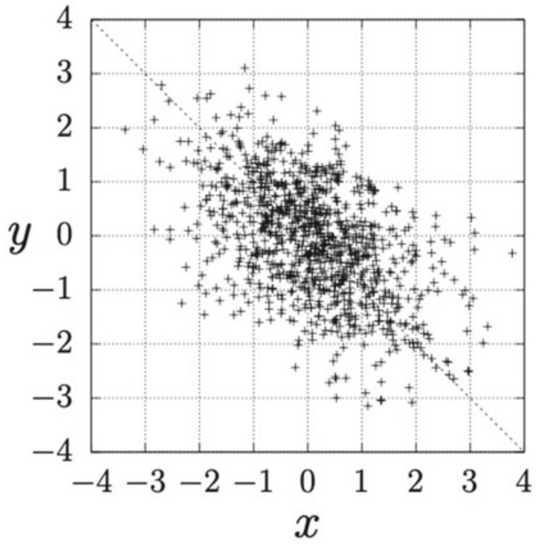

Fig. 4.14 A scatter plot of (x, y) that follows the bivariate Gaussian distribution $P(x, y) \propto e^{-\frac{x^2+y^2+xy}{2}}$. We used the Metropolis algorithm, with the step size 0.5 for both x and y. One configuration is sampled every 10 steps, and 10,000 configurations are collected in total

and output the configuration (x, y) every ten steps. We collected 1,000 configurations (10,000 steps) and plotted them in Fig. 4.14. (In this case, we know the probability distribution has the largest weight at $x = y = 0$. Therefore we set the initial configuration to be $x = y = 0$, to save the time for thermalization.) The dashed diagonal line is $y = -x$. The points (x, y) are distributed roughly along this line, and hence, we can confirm the tendency that "if x is a large positive value (good at mathematics) then y is a large negative value (bad at baseball)" and "if x is a large negative value (bad at mathematics) then y is a large positive value (good at baseball)". As a point (x, y) goes further from the center of the distribution ($(x, y) = (0, 0)$), the density becomes lower.

Now we have generated the probability distribution. As an application, let us design a life plan. Suppose some boys and girls do not know what they are good at, and they are wondering whether they should become a physicist or a baseball player. They would want to know the expectation value of their income as one of the factors for their career choice. For such an estimate, we need "salary functions" s_{physics} and s_{baseball} that relate the math and baseball skills to the salaries as a physicist or baseball player.

To become a physicist, it is better to be good at mathematics. However, even if one is not good at mathematics, somehow it is possible to survive and write papers. In this sense, some income is expected even if x is small. Another crucial fact is that, even if one is extremely good at mathematics, a physicist's salary cannot be huge. It is reasonable to assume that baseball skills do not affect a salary as a physicist, so we assume the salary function of physicists is a function of math skill x only and takes the following form:

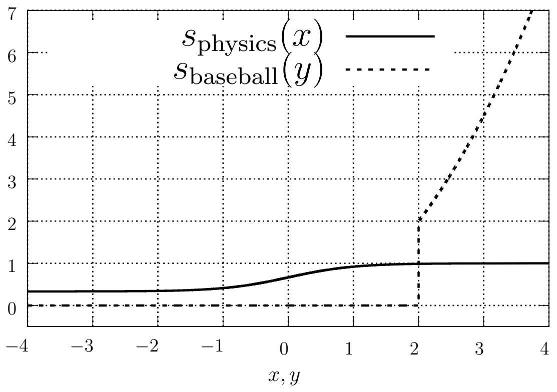

Fig. 4.15 Math skills x vs salary as physicist $s_{\text{physics}}(x)$ (solid line) and baseball skills y vs salary as baseball player $s_{\text{baseball}}(y)$ (dashed line)

$$s_{\text{physics}}(x) = \frac{2 + \tanh x}{3}. \tag{4.27}$$

The graph is shown in Fig. 4.15. Overall, this function is very smooth.

We expect that the salary function of baseball players s_{baseball} is very different. Unless one is exceptionally good at baseball, namely, unless y is sufficiently large, one cannot become a professional baseball player. Hence, the salary is zero below a certain threshold. However, if one actually becomes a professional player, their salary can be astronomical. Therefore, salary increases quickly beyond the threshold value of y. Probably, math skills do not affect the salary as a baseball player, so let us assume that s_{baseball} is a function of y only, and takes the following form:

$$s_{\text{baseball}}(y) = \begin{cases} 0 & (y \le 2) \\ \frac{y^2}{2} & (y > 2) \end{cases} \tag{4.28}$$

The graph is shown in Fig. 4.15, together with s_{physics}. A discontinuity at $y = 2$ is an important feature.

Let us calculate the expectation values $\langle s_{\text{physics}}(x)\rangle$ and $\langle s_{\text{baseball}}(y)\rangle$ by using the configurations generated via the Metropolis algorithm.

Let us see $\langle s_{\text{physics}}(x)\rangle$ first. Analytically, we can show that $\langle s_{\text{physics}}(x)\rangle = 2/3 = 0.66\ldots$. Can we reproduce this value via MCMC? We repeated the simulation 100 times with different sequences of the pseudorandom numbers, and calculated the average and standard deviation of 100 streams. The result is shown in Fig. 4.16. The horizontal axis is the number of samples K used to calculate the expectation value. We picked up four typical streams and showed them by purple lines. As K becomes large, the expectation value quickly converges to the exact analytic value. The convergence is quick because $s_{\text{physics}}(x)$ does not change violently with x (the difference is at most factor three) and all configurations give similar contributions. As a result, all configurations are effectively used and values close to the right answer can be obtained without using too many configurations.

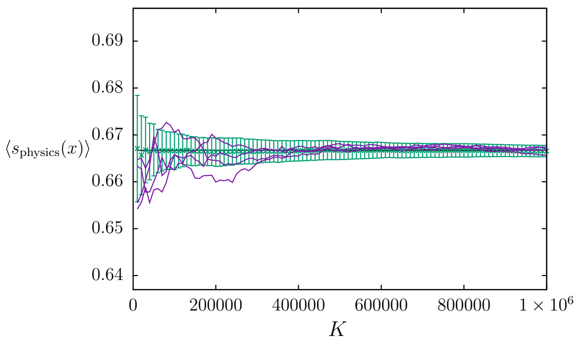

Fig. 4.16 The expectation value of $s_{\rm physics}(x)$ is calculated by using the Gaussian distribution $P(x, y) \propto e^{-\frac{x^2+y^2+xy}{2}}$. The horizontal axis is the number of configurations K used for the calculation. 100 independent simulations are performed by using different sequences of random numbers. The average and the standard deviation of the 100 streams are shown in green. The purple lines are four typical streams. Quick convergence to the right answer is observed

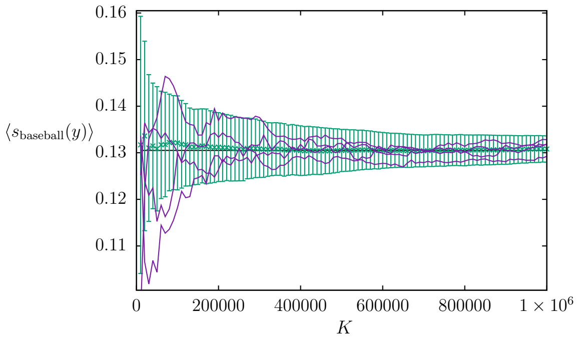

Fig. 4.17 The expectation value of $s_{\rm baseball}(y)$ is calculated by using the Gaussian distribution $P(x, y) \propto e^{-\frac{x^2+y^2+xy}{2}}$. As in Fig. 4.16, the horizontal axis is the number of configurations K used for the calculation. 100 independent simulations are performed by using different sequences of the random numbers. The average and the standard deviation of the 100 streams are shown in green. The purple lines are four typical streams. Because $s_{\rm baseball}(y)$ is an extreme function that has a big jump, the convergence to the right answer is rather slow and many configurations are needed for a precise estimate

Next, let us calculate $\langle s_{\rm baseball}(y)\rangle$. The answer is $\langle s_{\rm baseball}(y)\rangle = 0.1305\ldots$. Can we reproduce this value, as we did for $\langle s_{\rm physics}(x)\rangle$? The result is shown in Fig. 4.17, by using the same scale as Fig. 4.16. This plot looks very different from that for $\langle s_{\rm physics}(x)\rangle$; even with a fairly large-K, the expectation values from different streams differ significantly and the convergence to the right value is very slow. This is because rare configurations ($y > 2$) have large contributions while the majority of the configurations ($y \leq 2$) do not contribute at all. In other words, the peak of the probability distribution $P(x, y)$ and the configurations important for the expectation value do not overlap. If we calculate the expectation value in such a situation, wrong answers may be obtained unless we collect a very large number of configurations. Although, in the current case, we can increase the number of configurations by brute force, if the integrand is even more extreme (say, if only $y > 10$ contribute), or if the simulation is costly and it is hard to collect many configurations, some improvement is needed. Below, we will explain a method to solve this problem.

Solving the Overlap Problem

As some of you might have noticed, this is essentially the overlap problem explained in Sect. 4.5, and hence, can be resolved in the same manner. For example, let us define a new action by shifting the value of y by α, as

$$S(x, y; \alpha) = \frac{x^2 + (y-\alpha)^2 + x(y-\alpha)}{2}. \tag{4.29}$$

The partition function associated with $S(x, y; \alpha)$ is defined as

$$Z_\alpha = \int dx \int dy e^{-S(x,y;\alpha)}. \tag{4.30}$$

Then we can express $\langle s_{\text{baseball}}(y)\rangle$ in the following manner, by using $\{\alpha_i\}$ $(i = 1, \ldots, M)$:

$$\begin{aligned}\langle s_{\text{baseball}}(y)\rangle &= \frac{1}{Z_0}\int dx \int dy s_{\text{baseball}}(y) e^{-S(x,y;0)} \\ &= \frac{Z_{\alpha_1}}{Z_0}\cdot\frac{Z_{\alpha_2}}{Z_{\alpha_1}}\cdot\frac{Z_{\alpha_3}}{Z_{\alpha_2}}\cdots\frac{Z_{\alpha_M}}{Z_{\alpha_{M-1}}}\cdot\frac{1}{Z_{\alpha_M}}\int dx\int dy s_{\text{baseball}}(y)e^{-S(x,y;0)}.\end{aligned} \tag{4.31}$$

Let us denote the expectation value of a function $f(x, y)$ with the weight $e^{-S(x,y;\alpha)}$ as

$$\langle f(x,y)\rangle_\alpha = \frac{1}{Z_\alpha}\int dx \int dy f(x,y) e^{-S(x,y;\alpha)}. \tag{4.32}$$

Then, the partition functions can be written as

$$Z_{\alpha_{i+1}} = \int dx \int dy e^{-S(x,y;\alpha_{i+1})} = \int dx \int dy e^{-S(x,y;\alpha_{i+1})+S(x,y;\alpha_i)} e^{-S(x,y;\alpha_i)} \tag{4.33}$$

and hence the ratio is

$$\frac{Z_{\alpha_{i+1}}}{Z_{\alpha_i}} = \left\langle e^{-S(x,y;\alpha_{i+1})+S(x,y;\alpha_i)}\right\rangle_{\alpha_i} \equiv \left\langle e^{-\Delta_{i+1,i}}\right\rangle_{\alpha_i}. \tag{4.34}$$

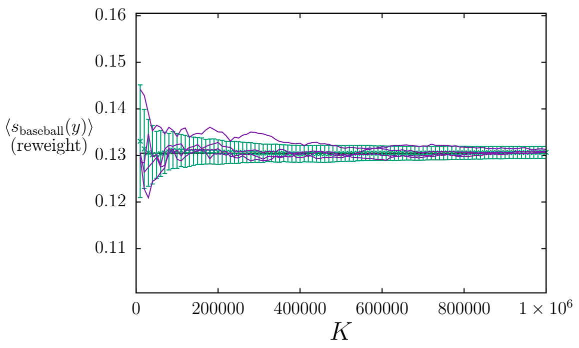

Fig. 4.18 The expectation value of $s_{\text{baseball}}(y)$ is calculated by using the reweighting method (4.35). We took $M = 2$, $\alpha_1 = 1.5$ and $\alpha_2 = 3$. As expected, the convergence to the right answer is fast

Here we used a shorthand notation $\Delta_{i+1,i} \equiv S(x, y; \alpha_{i+1}) - S(x, y; \alpha_i)$. Therefore, $\langle s_{\text{baseball}}(y)\rangle$ written in the form of (4.31) can be expressed as a product of the expectation values with the probability weights $e^{-S(x,y;\alpha_i)}$ $(i = 0, \ldots, M)$,

$$\langle s_{\text{baseball}}(y)\rangle = \left\langle e^{-\Delta_{1,0}}\right\rangle_{\alpha_0} \left\langle e^{-\Delta_{2,1}}\right\rangle_{\alpha_1} \cdots \left\langle e^{-\Delta_{M,M-1}}\right\rangle_{\alpha_{M-1}} \left\langle e^{-\Delta_{0,M}} s_{\text{baseball}}(y)\right\rangle_{\alpha_M} . \quad (4.35)$$

Here we set $\alpha_0 = 0$. The key point of this deformation is that, by choosing $\{\alpha_1, \ldots, \alpha_M\}$ appropriately, all expectation values $\left\langle e^{-\Delta_{i+1,i}}\right\rangle_{\alpha_i}$ and $\left\langle e^{-\Delta_{0,M}} s_{\text{baseball}}(y)\right\rangle_{\alpha_M}$ appearing in (4.35) can be calculated without encountering the overlap problem. To calculate $\left\langle e^{-\Delta_{i+1,i}}\right\rangle_{\alpha_i}$ efficiently, we just have to take α_i and α_{i+1} sufficiently close. To calculate the last term $\left\langle e^{-\Delta_{0,M}} s_{\text{baseball}}(y)\right\rangle_{\alpha_M}$, we should take α_M to be 2 or 3.

We applied this method with $M = 2$, $\alpha_1 = 1.5$, and $\alpha_2 = 3$. The result is shown in Fig. 4.18. Although we divided the task into only three pieces ($i = 0, 1, 2$), the convergence to the correct value became much quicker than the naive approach (Fig. 4.17). This is evidence that the overlap problem was resolved. It may not be a very sophisticated method, but that we use MCMC would mean we focus on the practical utility rather than the beauty, so we should not mind!

Now we could design a life plan. The expectation value of the salary as a physicist is about 0.667, while as a baseball player about 0.131 is expected. If the talent is unknown, the probability distribution used here and the salary functions are reasonable, and the salary is an important factor in life, then one should become a physicist rather than a baseball player. Note however that, for people who seriously care about salary, there are better jobs than a physicist.

Distributions with Many Variables

Even if there are a lot more variables, say $n = 100$, the same method can be used. It is better to update the variables one by one because the acceptance rate can become very small otherwise, unless the step size is very small.

When x_a is varied to $x_a + \Delta x_a$, the change of $S(x_1, \ldots, x_n) = \frac{1}{2}\sum_{i,j=1}^{n} A_{ij} x_i x_j$ can be calculated just by looking at the terms containing x_a:

$$\frac{1}{2} A_{aa} x_a x_a + \sum_{i \neq a} A_{ia} x_i x_a \tag{4.36}$$

Among $\frac{n(n+1)}{2}$ terms in $S(x_1, \ldots, x_n)$ (n terms for $i = j$ and $\frac{n(n-1)}{2}$ terms for $i < j$; no need for considering $i > j$ because $A_{ij} = A_{ji}$), there are only n terms that contain x_a, so the computational cost can be reduced drastically.

4.9 Exercises

1. In the main text, when $x' = x^{(k)} + \Delta x$ is proposed as a candidate of $x^{(k+1)}$, Δx was taken from the uniform random number. Actually, other kinds of random numbers can also be fine. Show that Δx can be chosen from the Gaussian random number. Show that, more generally, the detailed balance condition can be preserved as long as Δx and $-\Delta x$ appear with the same probability.
2. Show that Δx can be chosen from the uniform distributions with the step size $c = 1$ and $c = 100$ at the even and odd steps, respectively, as mentioned in Sect. 4.7.1. Is the detailed balance condition preserved?
3. Show that the step size can be chosen randomly from $c = 1, 2, 3, 4, 5$ or 6, with the probability $\frac{1}{6}$ at each step. (see Sect. 4.7.1.)
4. What if the variable is discrete? For example, if x takes only integer values, how should we choose Δx?
5. How can we estimate the error bar of the histogram of the probability distribution $P(x)$ obtained via the Markov Chain Monte Carlo simulation?
6. If the integral of $e^{-S(x)}$ is not finite, what happens in MCMC?
7. When we showed the detailed balance condition, we implicitly used the fact that the Jacobian is 1. (If a transformation $x \to x'$ maps $[x, x + \epsilon]$ to $[x', x', +\epsilon']$, the Jacobian is the ratio of the width, $\frac{\epsilon'}{\epsilon}$.) In the examples considered in this book, we can easily see that Jacobian is 1, unless otherwise stated. (For a little bit nontrivial case, see the HMC algorithm explained in Sect. 5.1.) What could be a problem if the Jacobian were not 1?
8. Show that both ⟨Update simultaneously⟩ and ⟨Update one by one⟩ are legitimate procedures.
9. Show that a different step size c_i can be used for each variable x_i.

Solutions

1. We generate Δx with the Gaussian weight $\frac{e^{-\frac{(\Delta x)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma}$. Then the transition probability $T(x \to x')$ is

$$T(x \to x') = \frac{e^{-\frac{(x-x')^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma} \times \min\left(1, e^{S(x)-S(x')}\right). \tag{4.37}$$

This is obtained by replacing $\frac{1}{2c}$ in (4.4) with $\frac{e^{-\frac{(x-x')^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma}$. The Eqs. (4.6) and (4.7) change in a similar manner. When $S(x) \geq S(x')$, we have

$$P(x) \cdot T(x \to x') = \frac{e^{-S(x)}}{Z} \times \frac{e^{-\frac{(x-x')^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma} \tag{4.38}$$

$$P(x') \cdot T(x' \to x) = \frac{e^{-S(x')}}{Z} \times \frac{e^{-\frac{(x'-x)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma} \times e^{S(x')-S(x)} = \frac{e^{-S(x)}}{Z} \times \frac{e^{-\frac{(x-x')^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma}. \tag{4.39}$$

In this way, we can show the detailed balance condition $P(x) \cdot T(x \to x') = P(x') \cdot T(x' \to x)$.
More generally, let $f(\Delta x)$ be the probability distribution of $f(\Delta x)$. Then, if $S(x) \geq S(x')$,

$$P(x) \cdot T(x \to x') = \frac{e^{-S(x)}}{Z} \times f(x'-x) \tag{4.40}$$

$$P(x') \cdot T(x' \to x) = \frac{e^{-S(x')}}{Z} \times f(x-x') \times e^{S(x')-S(x)} = \frac{e^{-S(x)}}{Z} \times f(x-x'). \tag{4.41}$$

Therefore, the detailed balance condition is satisfied as long as $f(x'-x) = f(x-x')$.

2. Because the step size alternates between $c = 1$ and $c = 100$, two different transition probabilities appear at even and odd steps. Therefore, to define the transition probability, we need a label to distinguish even and odd steps. Hence, let us introduce a variable y that takes two values, $y = 0$ (even) or $y = 1$ (odd), and specify the state by a pair (x, y). Then,

$$T((x,0) \to (x',1)) = T_{c=1}(x \to x'), \qquad T((x,1) \to (x',0)) = T_{c=100}(x \to x'). \tag{4.42}$$

It is easy to check that it is an irreducible Markov chain. However the value of y alternates as 0, 1, 0, 1, . . ., and hence the period is 2. Furthermore, the detailed balance does not hold; if we forget about y, then the following equalities hold:

$$P(x)T_{c=1}(x \to x') = P(x')T_{c=1}(x' \to x),$$
$$P(x)T_{c=100}(x \to x') = P(x')T_{c=100}(x' \to x). \tag{4.43}$$

However, if we take y into account then the actual relations are

$$P(x)T((x,0) \to (x',1)) = P(x')((x',0) \to (x,1)),$$
$$P(x)T((x,1) \to (x',0)) = P(x')((x',1) \to (x,0)), \tag{4.44}$$

which are slightly different from the detailed balance condition.
If we combine two steps $y = 0 \to 1 \to 0$ and regard it as one step, it is an irreducible, aperiodic Markov chain. Still, the combined transition function

$$T(x \to x'') = \int dx' T_{c=1}(x \to x')T_{c=100}(x' \to x'') \tag{4.45}$$

does not satisfy the detailed balance condition, i.e.,

$$P(x)T(x \to x') \neq P(x')T(x' \to x). \tag{4.46}$$

If we use a different transition probability

$$\tilde{T}(x \to x'') = \int dx' T_{c=100}(x \to x')T_{c=1}(x' \to x''), \tag{4.47}$$

then

$$P(x)T(x \to x') = P(x')\tilde{T}(x' \to x) \tag{4.48}$$

holds, but it is slightly different from the detailed balance condition.
That the detailed balance condition is not satisfied is not necessarily a bad news, because the detailed balance condition is just a sufficient condition for MCMC to work. What we really need is the equilibrium condition

$$P(x) = \int dx' P(x')T(x' \to x), \tag{4.49}$$

namely, there is no problem as long as $P(x)$ is stationary. This relation can be confirmed as follows. From (4.43), we can show that $P(x)$ is a stationary under the transition $T_c(x' \to x)$:

$$P(x) = \int dx' P(x') T_c(x' \to x). \tag{4.50}$$

By using it twice, we can show that $P(x)$ is a stationary under the transition $T(x' \to x)$:

$$\begin{aligned}\int dx' P(x') T(x' \to x) &= \int dx' P(x') \int dx'' T_{c=1}(x' \to x'') T_{c=100}(x'' \to x)\\ &= \int dx'' \left(\int dx' P(x') T_{c=1}(x' \to x'') \right) T_{c=100}(x'' \to x)\\ &= \int dx'' P(x'') T_{c=100}(x'' \to x)\\ &= P(x).\end{aligned} \tag{4.51}$$

Therefore, the distribution of $(x, y = 0)$ converges to $P(x)$. The same holds for $(x, y = 1)$ as well.
This is just a small technical issue that depends on the details of the setup. For example, if the step size changes as $c = 1, 1, 100, 1, 1, 100, \ldots$, then by combining three steps $c = 1$, $c = 100$, $c = 1$ and regarding it as one step, all four conditions including the detailed balance can be satisfied.

3. We can easily show that Δx and $-\Delta x$ appear with the same probability; e.g., the probability of $\Delta x = \pm 0.5$ is $\frac{1}{6} \sum_{c=1}^{6} \frac{1}{2c}$, that of $\Delta x = \pm 1.41$ is $\frac{1}{6} \sum_{c=2}^{6} \frac{1}{2c}$, and so on. Hence we can use the result of Exercise 1 without modification. Unlike Exercise 2, there is no subtlety in this case, and the detailed balance condition is satisfied almost trivially.
4. We only have to make Δx discrete.
5. We can use the Jackknife method. To make a histogram from K samples $x^{(1)}, x^{(2)}, \ldots, x^{(K)}$, we divide x to bins with width dx, count the number of samples in each bin, and normalize such that the integral becomes 1. If the number of samples in the i-th bin is n_i, the height of the histogram is $\rho_i = \frac{n_i}{K \cdot dx}$.
Let us divide the samples into n groups consisting of w samples. Let $\tilde{\rho}_i^{(l,w)}$ be the histogram calculated from the l-th group of samples. Then the Jackknife error at each bin is $\Delta_{w,i} = \sqrt{\frac{1}{n(n-1)} \sum_{l=1}^{n} \left(\tilde{\rho}_i^{(l,w)} - \rho_i \right)^2}$.
6. Because the "probability" and "expectation value" cannot be defined, MCMC is not applicable. Let us consider $S(x) = -x^2$ as an example. Because $S(x)$ is smaller when $|x|$ is larger, x diverges to $+\infty$ or $-\infty$.
7. To obtain a probability from a probability density, we have to multiply the width of an infinitesimal interval. Therefore, strictly speaking, we needed to multiply ϵ, ϵ', etc, in the expressions appeared in a proof of the detailed balance condition. We did not include them because, if Jacobian is 1, they are the same overall factor and could be ignored. When the Jacobian is not 1, we cannot ignore them. Depending on the choice of Δx, the detailed balance condition could be broken.

8. For the $\langle$Update simultaneously$\rangle$, we can easily show that it is a Markov chain that satisfies the irreducibility (unless the domain of integration splits into multiple islands), aperiodicity, and the detailed balance condition. The proof is almost identical to the one for the case of one variable.
   $\langle$Update one by one$\rangle$ is similar to the setup in Exercise 2. There were two kinds of transitions in Exercise 2, but now there are n of them. It is straightforward to see it is an irreducible Markov chain. If we update x_j when k is j modulo n ($j = 1, 2, \ldots, n$), then by regarding $j = 1 \to j = 2 \to \cdots \to j = n \to j = 1$ as one step, such a Markov chain is aperiodic. The detailed balance does not hold, but the equilibrium condition is still satisfied, which is enough for our purpose.
   If we change the method slightly such that one of n variables $x_1, \ldots, x_n$ is chosen randomly with a probability $\frac{1}{n}$ and updated, then all conditions hold including the aperiodicity and detailed balance.
9. There is no problem as long as Δx and $-\Delta x$ appear with the same probability. This condition is kept trivially even if we take different step sizes c_i for different variables.

References

1. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines. J. Chem. Phys. **21**(6), 1087–1092 (1953)
2. M. Rosenbluth, Proof of validity on Monte Carlo method for canonical averaging. Technical report, Los Alamos Scientific Lab., 1953; AIP Conference Proceedings (American Institute of Physics, 2003)
3. M. Matsumoto, T. Nishimura, Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8**(1), 30 (1998)
4. M.H. Quenouille, Approximate tests of correlation in time-series. J. R. Statist. Soc. Ser. B (Methodol.) **11**(1), 68–84 (1949)
5. M.H. Quenouille, Notes on bias in estimation. Biometrika **43**(3–4), 353–360 (1956)
6. J. Tukey, Bias and confidence in not-quite large sample. Ann. Math. Statist. **29**, 614 (1958)
7. M. Hanada, M. Honda, Y. Honma, J. Nishimura, S. Shiba, Y. Yoshida, Numerical studies of the Abjm theory for arbitrary n at arbitrary coupling constant. J. High Energy Phys. **2012**(5), 1–34 (2012)
8. S. Lawrence, Sign problems in quantum field theory: classical and quantum approaches. Ph.D. thesis (Maryland, 2020)
9. K. Nagata, Finite-density lattice QCD and sign problem: current status and open problems, arXiv:2108.12423 [hep-lat]
10. R. Feynman, Simulating physics with computers. Int. J. Theor. Phys. **21**, 467–488 (1982)
11. J. Preskill, Simulating quantum field theory with a quantum computer, in *PoS, LATTICE2018* (2018), p. 024
12. M. Troyer, U.-J. Wiese, Computational complexity and fundamental limitations to fermionic quantum Monte Carlo simulations. Phys. Rev. Lett. **94**(17), 170201 (2005)