4
Dynamical methods

Simple rejection methods for high-dimensional probability distributions have, in general, a low acceptance probability. So low, in fact, that it makes them unfeasible to sample most distributions of interest. In this chapter, we will develop a variant of the rejection method that allows us, finally, to sample almost any probability distribution using simple rules. The method is that of rejection *with repetition*. Of course, there is no free lunch, and there is an important price to be paid when using these methods: as they produce correlated (as opposed to independent) values of the random variables, it turns out that the error of the estimate of the integral or sum increases with N, the number of variables. We will see, though, that the increase of the error with N is moderate as compared to the decrease of the average acceptance probability would a simple rejection method be used.

4.1
Rejection with repetition: a simple case

Our goal here is, again, to devise methods to generate values x_k that can be used in a sampling method in order to compute a given integral:

$$I = \int dx\, f_{\hat{\mathbf{x}}}(x) G(x) \tag{4.1}$$

using the sample mean and variance, (2.11)-(2.12). For that purpose we need to generate values x_k of a random variable distributed according to the given pdf $f_{\hat{\mathbf{x}}}(x)$. We propose now a modification of the rejection method that we name rejection with repetition. The modification is such that every time a proposal is rejected, instead of proposing a new value, we simply return the value that had been generated in the previous step. We will explain exactly what we mean using an example. Imagine we want to sample values from the one-variable pdf:

$$f_{\hat{\mathbf{x}}}(x) = C \exp[-\frac{1}{2}x^2 - x^4]. \tag{4.2}$$

We know already how to do this using rejection methods: propose a value of x according to a Gaussian $\hat{\mathbf{G}}(0,1)$ distribution and accept it with probability $h(x) = e^{-x^4}$. We even wrote a short program to do this:

```
1 x=ran_g()
 if (ran_u().gt.exp(-x**4)) goto 1
```

The idea of the repetition is to avoid going back whenever a value is rejected at the k-th trial, instead the previous value of the random variable, the one obtained at the $(k-1)$-th trial is kept. This can be programmed as:

```
xp=ran_g()
if (ran_u().lt.exp(-xp**4)) x=xp
```

Here `xp` is the proposed value that we accept with probability `exp(-x**4)`. If the value is not accepted, we do not try to propose a new value, but the current value of x (the one obtained in the previous call) is not modified. We just need to take into account that, just in case the trial at the first initial step fails, the value of x should have been initialized following some distribution $f_{\hat{\mathbf{x}}_0}(x)$ (for example a Gaussian distribution), so we need to add, before the first attempt to generate a random number and only then, a line like:

```
x=ran_f0()
```

where `ran_f0()` returns a random variable distributed according to some distribution $f_{\hat{\mathbf{x}}_0}(x)$, or simply

```
x=x0
```

with `x0` some initial value. This corresponds to taking $f_{\hat{\mathbf{x}}_0}(x) = \delta(x - x_0)$.

How does the output of such a routine looks like? Since we are repeating values when rejecting the proposed value, a typical output is:

```
k        x_k
0 -0.38348165934483291
1 -0.38348165934483291
2 -0.38348165934483291
3 -0.38348165934483291
4 0.47884603175682383
5 0.47884603175682383
6 0.37469679082063412
7 0.35122027008152767
8 -0.44596168309186857
9 -0.44596168309186857
10 -0.44596168309186857
```

We do not need much knowledge of statistics to see that in this list some numbers are not independent, in fact they are equal! The question is whether we can still use this list of numbers in the estimation of the integral (4.1). Put in another way, is it true that the series of numbers x_k are still distributed according to $f_{\hat{\mathbf{x}}}(x)$ as given by (3.90)? As one value x_{k+1} might be equal to the previous value x_k, we need to find the probability distribution that governs the outcome of the k-th step of this process.

Let $\hat{\mathbf{x}}_k$ be the random variable obtained during the k-step. Its precise form will depend on the outcome of the acceptance process. As there are two options: accept/reject, we consider the acceptance step to be a Bernoulli variable $\hat{\mathbf{y}}$ that can take two possible values $\hat{\mathbf{y}} = 1$, accept, and $\hat{\mathbf{y}} = 0$, reject. Direct application of (1.135) leads to:

$$f_{\hat{\mathbf{x}}_k}(x) = f_{\hat{\mathbf{x}}_k}(x|\text{accept})p(\text{accept}) + f_{\hat{\mathbf{x}}_k}(x|\text{reject})p(\text{reject}), \tag{4.3}$$

being $p(\text{accept})$ and $p(\text{reject}) = 1 - p(\text{accept})$ the probabilities of acceptance and rejection, respectively, at step k. Now we can use the equivalent of (1.128) in the case of a discrete random variable $\hat{\mathbf{y}}$, to modify the first term of this sum:

$$f_{\hat{\mathbf{x}}_k}(x) = \text{Prob}(\text{accept}|x)g(x) + f_{\hat{\mathbf{x}}_k}(x|\text{reject})(1 - p(\text{accept})), \tag{4.4}$$

where we have used that $g(x)$ is the pdf of proposing the value x at step k. The probability of accepting a given value x is $\text{Prob}(\text{accept}|x) = h(x)$. If we reject, the pdf at step k is the same one valid at step $k-1$: $f_{\hat{\mathbf{x}}_k}(x|\text{reject}) = f_{\hat{\mathbf{x}}_{k-1}}(x)$. Finally, the acceptance probability is given by (3.92). This leads to:

$$f_{\hat{\mathbf{x}}_k}(x) = h(x)g(x) + f_{\hat{\mathbf{x}}_{k-1}}(x)\left[1 - \int_{-\infty}^{\infty} h(x)g(x)\,dx\right]. \tag{4.5}$$

This recursion relation gives $f_{\hat{\mathbf{x}}_k}(x)$ in terms of $f_{\hat{\mathbf{x}}_{k-1}}(x)$. We should not be confused by this simple linear recursion relation. The solution, in terms of the initial distribution $f_{\hat{\mathbf{x}}_0}(x)$ is

$$f_{\hat{\mathbf{x}}_k}(x) = (1-\epsilon)^n \left[f_{\hat{\mathbf{x}}_0} - \frac{h(x)g(x)}{\int_{-\infty}^{\infty} h(x)g(x)\,dx}\right] + \frac{h(x)g(x)}{\int_{-\infty}^{\infty} h(x)g(x)\,dx}, \tag{4.6}$$

where we have written ϵ as the average acceptance probability. This relation can be written in terms of the pdf $f_{\hat{\mathbf{x}}}(x)$ we want to sample from as given from 3.90:

$$f_{\hat{\mathbf{x}}_k}(x) = (1-\epsilon)^k \left[f_{\hat{\mathbf{x}}_0} - f_{\hat{\mathbf{x}}}(x)\right] + f_{\hat{\mathbf{x}}}(x). \tag{4.7}$$

Then the answer to the question: are all the x_k values distributed according to $f_{\hat{\mathbf{x}}}(x)$? is, generally, NO, as we can not conclude from this formula that $f_{\hat{\mathbf{x}}_k}(x) = f_{\hat{\mathbf{x}}}(x),\ \forall k$. However, if the initial numbers are already distributed according to $f_{\hat{\mathbf{x}}}(x)$, i.e. if $f_{\hat{\mathbf{x}}_0}(x) = f_{\hat{\mathbf{x}}}(x)$, then we find $f_{\hat{\mathbf{x}}_k}(x) = f_{\hat{\mathbf{x}}}(x),\ \forall k$ and the numbers we obtain by this procedure are indeed distributed according the desired distribution. If, on the other hand, the initial pdf $f_{\hat{\mathbf{x}}_0}(x)$ is not equal to the one we want to sample, the situation is not so bad either. According to (4.7), the difference between the actual distribution at step k, $f_{\hat{\mathbf{x}}_k}(x)$, and the one we want to sample, $f_{\hat{\mathbf{x}}}(x)$, monotonically decreases with k. In fact, as $0 < \epsilon \le 1$, we have the exact result, independently of the initial distribution $f_{\hat{\mathbf{x}}_0}$:

$$\lim_{k\to\infty} f_{\hat{\mathbf{x}}_k}(x) = f_{\hat{\mathbf{x}}}(x). \tag{4.8}$$

Imagine $\epsilon = 0.5$. After $k = 10$ steps the factor $(1-\epsilon)^k \approx 10^{-3}$. If $k = 100$, $(1-\epsilon)^k \approx 10^{-30}$, an absolutely negligible contribution in most cases. Even for a low acceptance probability $\epsilon = 10^{-2}$ it takes about $k = 2300$ steps to have $(1-\epsilon)^k \approx 10^{-10}$. Therefore, in order to ensure that the produced x_k values satisfy the desired distribution, we need to discard some steps at the beginning. This is the process called **thermalization**. How many steps M_0 we need to discard depends on the distance between the initial and the desired distributions and the average acceptance probability ϵ.

The reader will have noticed that the process of generation of the numbers x_k can be understood in terms of a Markov chain, since the result at step k depends only on the distribution at step $k-1$. It is easy to write the recursion relation 4.5 in the form of the recursion equation 1.140 for a homogeneous Markov chain

$$f_{\hat{\mathbf{x}}_n}(x) = \int_{-\infty}^{\infty} f(x|y) f_{\hat{\mathbf{x}}_{n-1}}(y)\, dy, \tag{4.9}$$

with a transition probability density function

$$f(x|y) = h(x) g_{\hat{\mathbf{x}}}(x) + \left[1 - \int_{-\infty}^{\infty} h(x) g_{\hat{\mathbf{x}}}(x)\, dx\right] \delta(x-y). \tag{4.10}$$

The two terms of the sum in the right hand side correspond to the two possibilities: acceptance of a proposed value or rejection.

4.2 Statistical errors

Remember the approximation consisting in replacing an average value by the sample mean,

$$I = \int_{-\infty}^{\infty} G(x) f_{\hat{\mathbf{x}}}(x)\, dx = \hat{\mu}_M[G] \pm \sigma[\hat{\mu}_M], \tag{4.11}$$

with

$$\hat{\mu}_M[G] = \frac{1}{M} \sum_{k=1}^{M} G(x_k). \tag{4.12}$$

As usual, as dictated by Chebycheff's theorem, the error of the sample mean is measured by its standard deviation. However, we can not use now the relation:

$$\sigma^2[\hat{\mu}_M] = \frac{\sigma^2[G]}{M} \approx \frac{\hat{\sigma}^2_M[G]}{M} \tag{4.13}$$

because the derivation of this formula assumes that all contributions to the sum in (4.12) are independent from each other and we know this is not the case for the rejection with repetition method. Let us get now an useful expression for the error in the case of correlated values. We assume that we have thermalized conveniently the algorithm by discarding the necessary steps at the beginning and the Markov chain is in the steady state and, consequently, all variables x_k are distributed according to the same pdf $f_{\hat{\mathbf{x}}}(x)$.

The variance of the sample mean is defined as:

$$\sigma^2[\hat{\mu}_M] = \langle (\hat{\mu}_M - I)^2 \rangle, \tag{4.14}$$

replacing $\hat{\mu}_M$ from (4.12) and writing G_k instead of $G(x_k)$ for brevity in the notation, we obtain:

$$\begin{aligned} \sigma^2[\hat{\mu}_M] &= \left\langle \left[\frac{1}{M} \sum_{k=1}^{M} (G_k - I) \right]^2 \right\rangle \\ &= \frac{1}{M^2} \sum_{k=1}^{M} \sum_{j=1}^{M} \left\langle (G_k - I)(G_j - I) \right\rangle . \end{aligned} \tag{4.15}$$

In this double sum, we consider separately the terms with (i) $k = j$, (ii) $k > j$ and (iii) $j > k$. Due to the symmetry between k and j, the last two contributions are equal and we can write

$$\sigma^2[\hat{\mu}_M] = \frac{1}{M^2}\sum_{k=1}^{N}\langle(G_k - I)^2\rangle + \frac{2}{M^2}\sum_{k=1}^{M-1}\sum_{j=k+1}^{M}\langle(G_k - I)(G_j - I)\rangle. \tag{4.16}$$

Expanding the product in the second sum and using that $\langle G_i\rangle = I,\ \forall i$:

$$\begin{aligned}\sigma^2[\hat{\mu}_M] &= \frac{1}{M^2}\sum_{k=1}^{M}\left[\langle G_k^2\rangle - I^2\right] + \frac{2}{M^2}\sum_{k=1}^{M-1}\sum_{j=k+1}^{M}\left[\langle G_k G_j\rangle - I^2\right] \\ &= \frac{\sigma^2[G]}{M}\left[1 + \frac{2}{M}\sum_{k=1}^{M-1}\sum_{j=k+1}^{M}\rho_G(k,j)\right], \end{aligned}\tag{4.17}$$

we have used that all G_k yield the same variance:

$$\sigma^2[G] = \langle G_i^2\rangle - I^2, \tag{4.18}$$

and the definition of the normalized correlation between G_k and G_j:

$$\rho_G(k,j) = \frac{\langle G_k G_j\rangle - I^2}{\sigma^2[G]}. \tag{4.19}$$

Since we are in the steady state of a homogeneous Markov chain, the correlation function depends only on the difference $j - k \equiv i$, $\rho_G(k,j) = \rho_G(j-k) = \rho_G(i)$, and we can write:

$$\sigma^2[\hat{\mu}_M] = \frac{\sigma^2[G]}{M}\left[1 + \frac{2}{M}\sum_{k=1}^{M-1}\sum_{i=1}^{M-k}\rho_G(i)\right]. \tag{4.20}$$

Exchanging the summation order:

$$\sum_{k=1}^{M-1}\sum_{i=1}^{M-k}\rho_G(i) = \sum_{i=1}^{M-1}\sum_{k=1}^{M-i}\rho_G(i) = \sum_{i=1}^{M-1}(M-i)\rho_G(i), \tag{4.21}$$

since the sum over k could be performed as $\rho_G(i)$ does not depend on k. We have now the final result:

$$\sigma^2[\hat{\mu}_M] = \frac{\sigma^2[G]}{M}\left[2\sum_{i=1}^{M-1}\left(1 - \frac{i}{M}\right)\rho_G(i) + 1\right] \tag{4.22}$$

Let us now define the autocorrelation time τ_G as

$$\tau_G = \sum_{i=1}^{M-1}\left(1 - \frac{i}{M}\right)\rho_G(i), \tag{4.23}$$

we obtain the final expression for the variance of the sample mean, again replacing the true variance $\sigma^2[G]$ by the sample variance $\hat{\sigma}_M^2[G]$

$$\sigma^2[\hat{\mu}_M] = \frac{\hat{\sigma}_M^2[G]}{M}(2\tau_G + 1). \tag{4.24}$$

Let us stress that the only assumption in this definition is that the Markov chain is in the steady state where variables at each step k have the same distribution and correlations between steps k and j depend only of the difference $j-k$. For instance, if all variables $\hat{\mathbf{x}}_k$ were independent we would have $\rho_G(i) = \delta_{i,0}$, the autocorrelation time would be 0 and the variance would simplify to the known result $\sigma^2[\hat{\mu}_M] = \sigma^2[G]/M$. In general, the autocorrelation function $\rho_G(i)$ decays with i and tends to 0 as $i \to \infty$ (usually, but not always, the decay is exponential).

The autocorrelation time τ_G is a measure of the decay of the correlation function. It can be defined, loosely speaking, as the number of steps that are needed for the correlation to decay significantly from the initial value $\rho_G(0) = 1$. A more precise definition is simply (4.23). For instance, if the decay is truly exponential $\rho_G(i) = [\rho_G(1)]^i$, with $\rho_G(1) < 1$, then according with the definition (4.23) the correlation time is:

$$\tau_G = \sum_{i=1}^{M-1} \left(1 - \frac{i}{M}\right) [\rho_G(1)]^i = \frac{\rho_G(1)}{1-\rho_G(1)} - \frac{\rho_G(1)(1-[\rho_G(1)]^M)}{(1-\rho_G(1))^2 M}, \tag{4.25}$$

that can be reduced to

$$\tau_G = \frac{\rho_G(1)}{1-\rho_G(1)} \tag{4.26}$$

in the limit of large M. Indeed, in most cases, the correlation function decays rapidly and, in the limit of large M we can neglect the factor i/M in the definition of τ_G and write:

$$\tau_G = \sum_{i=1}^{\infty} \rho_G(i). \tag{4.27}$$

For the rejection with repetition method explained above, it is possible to compute the correlation function. If the acceptance probability is ϵ then either $G_{i+k} = G_k$ with probability $(1-\epsilon)^i$ (all i steps in going from k to $i+k$ have been rejections) or G_{i+k} is independent from G_k with probability $1-(1-\epsilon)^i$. This gives:

$$\langle G_k G_{i+k} \rangle = (1-\epsilon)^i \langle G_k^2 \rangle + [1-(1-\epsilon)^i] \langle G_k \rangle \langle G_{i+k} \rangle, \tag{4.28}$$

which leads to $\rho_G(i) = (1-\epsilon)^i$, an exponential decay. The autocorrelation time is independent of the function G and is given by $\tau_G = \frac{1-\epsilon}{\epsilon}$, then $2\tau_G + 1 = \frac{2-\epsilon}{\epsilon}$ and the correct expression for the estimator and its error is:

$$I = \hat{\mu}_M[G] \pm \frac{\hat{\sigma}_M[G]}{\sqrt{M}} \sqrt{\frac{2-\epsilon}{\epsilon}} \tag{4.29}$$

Note that this is equivalent to have used an effective number $M_{\text{eff}} = M\frac{\epsilon}{2-\epsilon} \leq M$ of independent contributions to the estimator. If we wanted to have the same error using a rejection method without repetition (i.e. trying again if the proposed value is rejected), then we would need an average number M_{eff}/ϵ of proposals. As $\frac{2-\epsilon}{\epsilon} \geq \frac{1}{\epsilon}$, it turns out that the method of rejection with repetition needs to generate more random numbers in order to yield the same statistical error. However, we will see in the next section that this method is the one that can be generalized to high-dimensional distributions while keeping a reasonable acceptance probability.

4.3 Dynamical methods

Rejection methods, independently on whether we use repetition or not, can have a very low acceptance probability. This is specially true if the proposal probability $g(x)$ is very distant from $f_{\hat{\mathbf{x}}}(x)$ and does not reflect the strong variations in value that $f_{\hat{\mathbf{x}}}(x)$ might have, i.e. the ratio $f_{\hat{\mathbf{x}}}(x_{\max})/f_{\hat{\mathbf{x}}}(x_{\min})$. This usually occurs when we have no clue on how the pdf $f_{\hat{\mathbf{x}}}(x)$ we want to sample looks like, a problem particularly severe and common for distributions taking values in a high N-dimensional space, $x = (x_1, \dots, x_N)$.

Let us give an specific example. Consider the distribution:

$$f_{\hat{\mathbf{x}}}(x_1, \dots, x_N) = \begin{cases} C \exp\left(- \sum_{i=1,j=i}^{N} x_i x_j \right), & \text{if } x_i \in (-1,1), \\ 0, & \text{if } x_i \notin (-1,1). \end{cases} \tag{4.30}$$

One might be tempted to use a rejection method in which the proposal are independent values $x \equiv (x_1, \dots, x_N)$ each one drawn from a $\hat{\mathbf{U}}(-1,1)$ or uniformly distributed in the interval $(-1,1)$. This proposal would be accepted with a probability $h(x_1, \dots, x_N)$ proportional to $f_{\hat{\mathbf{x}}}(x_1, \dots, x_N)$. We note that the maximum value of the distribution $f(x_1, \dots, x_N)$ occurs for $x_i = 0, \forall i$ and there are two equivalent minima for $x_i = 1, \forall i$ and $x_i = -1, \forall i$[1]. As there are a total of $N(N+1)/2$ terms in the sums of the exponential defining $f_{\hat{\mathbf{x}}}(x)$, the ratio between these two extreme values is $f_{\hat{\mathbf{x}}}(x_{\max})/f_{\hat{\mathbf{x}}}(x_{\min}) = \mathrm{e}^{N(N+1)/2}$. The optimal acceptance probability would be $h(x_1, \dots, x_N) = \exp\left(-\sum_{i=1,j=i}^{N} x_i x_j\right)$ whose minimum and maximum values are $e^{-N(N+1)/2}$ and 1, respectively. When N is large, it turns out that most of the proposed values will belong to a region in which the acceptance probability is very small. Hence the average acceptance probability is very small. This can be checked with the program:

```
program average
implicit none
integer, parameter :: n=100
integer m,ijk,i,j
double precision :: x(n)
double precision :: averh,sum,ran_u
m=1000000
averh=0.0d0
do ijk=1,m
 do i=1,n
   x(i)=2.d0*ran_u()-1.0d0
 enddo
 sum=0.0d0
 do i=1,n
 do j=i,n
```

1) It might help to derive these results the identity $\sum_{i=1,j=i}^{N} x_i x_j = \frac{1}{2}\left(\left(\sum_{i=1}^{N} x_i\right)^2 + \sum_{i=1}^{N} x_i^2\right)$.

```
 sum=sum+x(i)*x(j)
 enddo
 enddo
averh=averh+dexp(-sum)
enddo
write(6,*) n,averh/m
end program average
```

Running this program with different values of N we obtain that the average acceptance probability ϵ decreases with N. In fact, it is found that the average acceptance probability decays exponentially with N (see problem 1). For example, for $N = 10$ it is $\epsilon \approx 0.105$; for $N = 20$, $\epsilon = 1.68 \times 10^{-2}$; and for $N = 100$, $\epsilon \approx 3 \times 10^{-8}$, which means that we have to propose, on average, around 30 million numbers, in order to accept one, a very inefficient procedure, indeed.

How can we avoid proposing values which have a very low acceptance probability? The idea of the so-called dynamical methods is to use the information gathered at previous proposals to make a new proposal. In this way, when by repeated trials we reach the region in which the pdf is high, our proposals will not tend to abandon this region. Let us explain these ideas using yet a simpler example than the N-dimensional distribution 4.30. Consider the cut-off Gaussian probability distribution

$$f_{\hat{\mathbf{x}}}(x) = C \exp\left(-x^2/2\sigma^2\right), \quad x \in (-1,1) \tag{4.31}$$

for small values of the parameter σ[2]. If we propose "blindly" values uniformly distributed according to a uniform distribution $g(x)$ in the interval $(-1, 1)$ and accept this value with a probability $h(x) = \exp\left(-x^2/2\sigma^2\right)$, most of the times we are proposing values which have a very small acceptance probability[3]. However, when we reach a value near $x = 0$ and note that there is a high acceptance probability, what we should do is that the new proposed value is again close to 0, as we know now (and did not know before) that $x = 0$ is the region of high probability. If x_k is the value obtained at the k-th step, then the new proposal at step $k + 1$ could be, for example, a number x_{k+1} chosen uniformly in the interval $x_{k+1} \in (x_k - \delta, x_k + \delta)$ with δ a number comparable to σ, so we do not leave the region of large probability. This is the basic principle behind the dynamical methods.

Summing up, in the static methods explained in chapter 3, the new value for the random variable was chosen each time independently on previous values, whether in the dynamical methods the proposal (and the acceptance probability as we will see) depends on the previous values of the random variable. In other words, in the static methods, if we denote by x_n the value of the random variable generated at the n-th step, then both the proposal $g(x)$ and the acceptance $h(x)$ are independent on previous values $(x_1, \dots, x_{n-1})$. As a consequence, the pdf $f_{\hat{\mathbf{x}}_n}(x)$ at step n, was independent on the previous steps. Being more precise, in the static methods:

$$f_{\hat{\mathbf{x}}_n}(x_n|x_0, x_1, \dots, x_{n-1}) = f_{\hat{\mathbf{x}}_n}(x_n) = f_{\hat{\mathbf{x}}}(x_n), \quad n = 0, 1, 2, \dots \tag{4.32}$$

2) σ is not the rms, as the distribution does not extend for all real values of x.
3) The average acceptace probability is $\epsilon = \sqrt{\frac{\pi}{2}}\sigma\,\mathrm{erf}(1/\sigma\sqrt{2})$. This tends to $\epsilon \to \sqrt{\frac{\pi}{2}}\sigma$ as $\sigma \to 0$.

and all random variables x_n are independently distributed according to the same pdf $f_{\hat{\mathbf{x}}}(x)$.

In a dynamical method, both the proposal and the acceptance probability depend on the previous results. As a consequence, the pdf of the random variable x_n obtained in the n-th proposal/acceptance step depends on the previous values of the random variables $x_0, x_1, \dots, x_{n-1}$. The simplest way to implement this dependence is via the latest result, i.e. take

$$f_{\hat{\mathbf{x}}_n}(x_n|x_0, \dots, x_{n-1}) = f_{\hat{\mathbf{x}}_n}(x_n|x_{n-1}), \tag{4.33}$$

defining our series of proposal/acceptance steps as a Markov chain. Again, the simplest implementation uses a homogeneous Markov chain where the conditional probabilities are independent on the step, or $f_{\hat{\mathbf{x}}_n}(x_n|x_{n-1}) = f(x_n|x_{n-1})$.

Let us now go into the details of a dynamical method. We generate a realization of the Markov chain generating values $x_0, x_1, \dots, x_n, \dots$ of the random variables . The value x_n is sampled from a distribution $f(x_n|x_{n-1})$. Ideally, we would like all probability distributions $f_{\hat{\mathbf{x}}_n}(x)$ to be equal to the desired pdf $f_{\hat{\mathbf{x}}}(x)$. However, our discussion of the simple dynamical method of rejection with repetition tells us that it might be more reasonable to demand only that the asymptotic distribution reproduces the desired pdf, $\lim_{n\to\infty} f_{\hat{\mathbf{x}}_n}(x) = f_{\hat{\mathbf{x}}}(x)$. Our discussion about Markov chains in section 1.9 tells us that a sufficient condition for this to happen is the detailed balance condition:

$$f(y|x) f_{\hat{\mathbf{x}}}(x) = f(x|y) f_{\hat{\mathbf{x}}}(y). \tag{4.34}$$

Usually (but not always), the generation of random variables distributed according to $f(x|y)$ uses a rejection method. This means that we propose a value x from a pdf $g(x|y)$ and then accept it with a probability $h(x|y)$. As explained, both proposal and acceptance of a number x at step n depend on the value y obtained in the previous step. We now obtain the transition probability $f(x|y)$ that corresponds to this proposal/acceptance algorithm in the case of rejection with repetition. i.e. if the value x is not accepted, then we keep the previous value y. It will be clear in a moment why, of all possible rejection methods, we adopt this one. The derivation of the transition pdf is similar to the one leading to 4.10, but now we have to take into account that both the proposal pdf $g(x|y)$ and the acceptance probability $h(x|y)$ depend on y.

The transition pdf $f(x|y)$ has two contributions corresponding to the acceptance or not of the proposed value. Given a value y, the overall rejection probability is

$$\begin{aligned} P(\text{rejection}|y) &= 1 - P(\text{acceptance}|y) \\ &= 1 - \int h(z|y) g(z|y)\, dz \equiv 1 - \epsilon(y), \end{aligned} \tag{4.35}$$

where we have used (3.92) with the only modification that both the proposal and the acceptance probabilities now depend on the value y. If the proposal x is not accepted, then the transition probability is the Dirac-delta $\delta(x-y)$, as we keep the old value y. This leads to a transition probability:

$$f(x|y) = h(x|y) g(x|y) + \delta(x-y)\left[1 - \epsilon(y)\right]. \tag{4.36}$$

Similarly, we have:

$$f(y|x) = h(y|x)g(y|x) + \delta(x-y)\left[1-\epsilon(x)\right]. \tag{4.37}$$

Which functions $h(x|y)$ and $g(x|y)$ can be used? The only requirement is that the stationary pdf of this Markov chain is the distribution $f_{\hat{\mathbf{x}}}(x)$. We enforce this by imposing the sufficient condition of detailed balance[4]. Replacing $f(x|y)$ and $f(y|x)$ in the detailed balance condition (4.34), the two terms with delta-functions are identical as they force $x = y$, leading to

$$h(y|x)g(y|x)f_{\hat{\mathbf{x}}}(x) = h(x|y)g(x|y)f_{\hat{\mathbf{x}}}(y), \tag{4.38}$$

as the equation to be satisfied by the proposal $g(x|y)$ and the acceptance $h(x|y)$ to ensure that the stationary distribution of the Markov chain is indeed $f_{\hat{\mathbf{x}}}(x)$.

It is essential to return the value $x_n = x_{n-1}$ if the proposal x is not accepted instead of keep on proposing new values until they get accepted as we used to do in a standard rejection method. The reason is simple to understand. If we sampled $f(x|y)$ without repetition, i.e. trying new values until one is accepted, the same argument that led to the distribution (3.90) tells us that the resulting pdf is

$$f(x|y) = \frac{h(x|y)g(x|y)}{\int dz\; h(z|y)g(z|y)}, \tag{4.39}$$

and similarly

$$f(y|x) = \frac{h(y|x)g(y|x)}{\int dz\; h(z|x)g(z|x)}. \tag{4.40}$$

The detailed balance condition to be satisfied by the proposal $g(x|y)$ and the acceptance $h(x|y)$ probabilities is

$$\frac{f_{\hat{\mathbf{x}}}(y)h(x|y)g(x|y)}{\int dz\; h(z|y)g(z|y)} = \frac{f_{\hat{\mathbf{x}}}(x)h(y|x)g(y|x)}{\int dz\; h(z|x)g(z|x)}, \tag{4.41}$$

and, due to the integrals, it is difficult to give general solutions for $g(x|y)$ and $h(x|y)$ satisfying this equation.

We write now in full the recurrence relation of the Markov chain:

$$\begin{aligned} f_{\hat{\mathbf{x}}_{n+1}}(x) &= \int f(x|y)f_{\hat{\mathbf{x}}_n}(y)\,dy \qquad (4.42)\\ &= \int h(x|y)g(x|y)f_{\hat{\mathbf{x}}_n}(y)\,dy + f_{\hat{\mathbf{x}}_n}(x)\left[1-\int dz h(z|x)g(z|x)\right] \end{aligned}$$

where

$$\epsilon(x) = \int dz\; h(z|x)g(z|x) \tag{4.43}$$

is the acceptance probability given x. The overall average acceptance probability is

$$\epsilon = \langle \epsilon(x) \rangle = \int dx\; \epsilon(x) f_{\hat{\mathbf{x}}}(x). \tag{4.44}$$

We will review now different solutions to the detailed balance condition in its form (4.38). There are, mainly, two ways to find the solution of this equation: the Metropolis *et al.* algorithm and the heat-bath algorithm.

4) The authors are not aware of any solution which does not use the detailed balance condition.

4.4 Metropolis *et al.* algorithm

In the Metropolis *et al.* algorithm we fix the proposal distribution $g(x|y)$ and solve for the acceptance probability. We must find then a function $h(x|y)$ satisfying:

$$0 \le h(x|y) \le 1 \tag{4.45}$$

$$\frac{h(x|y)}{h(y|x)} = q(x|y) \tag{4.46}$$

where we have introduced

$$q(x|y) = \frac{g(y|x) f_{\hat{\mathbf{x}}}(x)}{g(x|y) f_{\hat{\mathbf{x}}}(y)}, \tag{4.47}$$

which satisfies the condition:

$$q(y|x) = \frac{1}{q(x|y)}. \tag{4.48}$$

All solutions of (4.46) can be found introducing the function $\omega(z)$ by means of

$$h(x|y) = \sqrt{q(x|y)}\omega(x,y), \tag{4.49}$$

which replaced in (4.46) requires the symmetry condition $\omega(x,y) = \omega(y,x)$. Condition (4.45) is more delicate. A possibility is to demand that $\omega(x,y)$ depends on x and y through the function $q(x|y)$: $\omega(x,y) = \omega(q(x|y))$, with the symmetry condition $\omega(x,y) = \omega(y,x)$ being equivalent to $\omega(q) = \omega(1/q)$. The requirement (4.45) then leads to $0 \le \sqrt{q}\omega(q) \le 1$ or $\omega(q) \le 1/\sqrt{q}$ from where $\omega(q) = \omega(1/q) \le 1/\sqrt{1/q}$. We are looking, then, for functions $\omega(q)$ satisfying:

$$\omega(q) = \omega(1/q), \tag{4.50}$$

$$\omega(q) \le \sqrt{q}. \tag{4.51}$$

The first solution is the one given by Metropolis *et al.*:

$$\omega(q) = \min\left(q, 1/q\right). \tag{4.52}$$

It satisfies trivially $\omega(q) = \omega(1/q)$. For condition (4.51) we take separately the cases $q \le 1$ and $q \ge 1$. If $q \le 1$ then $\omega(q) = q \le \sqrt{q}$. For $q \le 1$, it is $\omega(q) = 1/\sqrt{q} \le \sqrt{q}$. This leads to one of the most widely used solutions of the detailed balance condition:

$$h(x|y) = \min\left(1, q(x|y)\right). \tag{4.53}$$

The second most widely used solution is that of Glauber,

$$\omega(q) = \frac{1}{\sqrt{q} + \sqrt{1/q}}, \tag{4.54}$$

which trivially satisfies (4.50)-(4.51). In terms of the transition probability, we have:

$$h(x|y) = \frac{q(x|y)}{1 + q(x|y)}. \tag{4.55}$$

Another solution was used later by van Beijeren and Schulman. It is simply $\omega(q) = C$, constant. The constant has to be chosen such that $C \le \max_{x,y} \sqrt{q(x|y)}$, in order to fulfill the condition $h(x|y) \in [0,1]$.

Let us now apply the Metropolis *et al.* algorithm to some specific examples.

Gaussian distribution We consider a random variable $\hat{\mathbf{x}}$ whose pdf is Gaussian:

$$f_{\hat{\mathbf{x}}}(x) = A\exp(-\frac{x^2}{2}). \tag{4.56}$$

The normalization constant C is irrelevant for the Metropolis algorithm. We take the following proposal probability:

$$g(x|y) = \begin{cases} \frac{1}{2\Delta}, & |x-y| < \Delta, \\ 0, & \text{otherwise.} \end{cases} \tag{4.57}$$

In other words, x is drawn from a uniform distribution in the interval $(y-\Delta, y+\Delta)$. The constant Δ is arbitrary for now, but later we will determine which is the best choice for it. Obviously, the proposal satisfies the symmetry condition

$$g(x|y) = g(y|x) \tag{4.58}$$

so the function $(x|y)$ is

$$q(x|y) = \frac{g(y|x)f_{\hat{\mathbf{x}}}(x)}{g(x|y)f_{\hat{\mathbf{x}}}(y)} = \exp(\frac{y^2-x^2}{2}). \tag{4.59}$$

We now need an acceptance probability $h(x|y)$. We will use Metropolis *et al.* choice $h(x|y) = \min(1, q(x|y))$. In our example, it is:

$$q(x|y) = \begin{cases} 1, & |y| > |x|, \\ \exp(\frac{y^2-x^2}{2}), & |y| \le |x|. \end{cases} \tag{4.60}$$

As usual, the acceptance step is a Bernoulli process of probability q. We draw a $\hat{\mathbf{U}}(0,1)$ random number u and compare it with q. If $u \le q$ we accept the proposal. Note that if $q = 1$, the proposal is always accepted and there is no need to spend time drawing a random number and comparing it to 1. The algorithm can be summarized as follows:

1) Given y propose x uniformly from $(y-\Delta, y+\Delta)$.
2) If $|x| \le |y|$, accept x.
3) If $|x| > |y|$, accept x with probability $q = \exp(\frac{y^2-x^2}{2})$. If the value x is rejected, return y as the value of the Gaussian variable.

The program could look like this:

```
double precision function ran_gauss_mc(y,delta)
x=y+delta*(2.0d0*ran_u()-1.0d0)
if(abs(x).gt.abs(y)) then
 if (ran_u().gt.exp(0.5d0*(y-x)*(y+x))) then
! Reject. Do not accept the proposed value. Keep old one.
```

```
 ran_gauss_mc=y
 return
 endif
endif
y=x
ran_gauss_mc=y
end function ran_gauss_mc
```

Before the first call to this routine we need to set an initial value for y, for instance, $y = 0$.

We can understand intuitively the way the algorithm works. If the proposal tends to increase he probability, i.e. if $f_{\hat{\mathbf{x}}}(x) \ge f_{\hat{\mathbf{x}}}(y)$, then the new value is accepted. This moves the Markov chain towards values of high probability. Still, if $f_{\hat{\mathbf{x}}}(x) < f_{\hat{\mathbf{x}}}(y)$, the new value is not systematically rejected, so there is a chance of visiting regions of low probability. As we have shown, the resulting distribution of values of x coming from these steps (unconditional acceptance of values with larger probability, and conditional acceptance otherwise) is precisely $f_{\hat{\mathbf{x}}}(x)$.

There is still the issue of the parameter Δ. Notice, first, that whatever the value of Δ, the proposal $g(x|y)$ given by (4.57) leads to an ergodic algorithm because it allows each and every real value x to be proposed eventually and it can not be trapped in loops. It is a random walk in the real space that does not have any forbidden region. Certainly, some values will be accepted more often that others, but the proposal probability is such that all values of x could be visited at one time or another. Note that the average acceptance probability ϵ depends on Δ. Using (4.44), and after a lengthy calculation, we obtain:

$$\epsilon = 1 - \mathrm{erf}\left(\frac{\Delta}{2\sqrt{2}}\right) + \frac{4}{\Delta\sqrt{2\pi}}\left(1 - e^{-\Delta^2/8}\right), \tag{4.61}$$

which monotonically decreases to 0 as Δ increases from its maximum value $\epsilon = 1$ at $\Delta = 0$. Based on this result, one might think, wrongly, that the smaller Δ the more efficient the algorithm is, as the acceptance probability is larger. However, it is clear that for very small Δ, the proposed value x will be very close to the old value y, and hence there will be a large correlation between x and y, the old and the proposed (and very likely to be accepted) value. This induces a large correlation time and an increase in the errors. Instead, we come to the general idea that in order to optimize the algorithm, the parameter Δ has to be chosen such that the correlation time τ_G (and hence the statistical errors) reaches its minimum value. We will come to this important point later.

Here we can check what would happen if we did not repeat the previous value in case of rejection and insisted in proposing a new value until it got accepted, as we used to do in a standard rejection method.

```
double precision function ran_gauss_mc_bad(y,delta)
1 x=y+delta*(2*ran_u()-1)
if(abs(x).gt.abs(y)) then
 if (ran_u().gt.exp(0.5d0*(y-x)*(y+x))) goto 1
endif
y=x
```

```
ran_gauss_mc_bad=y
end function ran_gauss_mc_bad
```

The reader can check that tis routine does not produce numbers distributed according to a Gaussian distribution (it becomes worse as Δ increases).

Poisson distribution We give an example now of the application of the Metropolis *et al.* algorithm to a discrete distribution. We consider a Poisson random variable that takes integer values $\hat{\mathbf{x}} = k,\ k = 0, 1, 2, 3, \dots$ with probability

$$p_k = e^{-\lambda}\frac{\lambda^k}{k!}. \tag{4.62}$$

We shall proceed very similarly to the Gaussian distribution. Given a current value k we must propose a new value k' sampled from a distribution $g(k'|k)$. We propose to use:

$$g(k'|k) = \begin{cases} 1/2, & k' = k+1, \\ 1/2, & k' = k-1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.63}$$

We only propose values k that differ from k in one unit. In this way, we obtain an ergodic algorithm: all integer values have, in principle, the chance to be proposed at one time or another. Note, furthermore, that this proposal satisfies the symmetry condition $g(k'|k) = g(k|k')$.

With the choice (4.63), function $q(k'|k)$

$$q(k'|k) = \frac{p_{k'}}{p_k} = \lambda^{k'-k}\frac{k!}{k'!}. \tag{4.64}$$

And the acceptance probability $h(k'|k) = \min(1, q(k'|k)$ can take two possible values:
(a) If $k' = k+1$ then:

$$q(k+1|k) = \frac{\lambda}{k+1} \longrightarrow h(k+1|k) = \begin{cases} 1, & \lambda \geq k+1, \\ \frac{\lambda}{k+1}, & \lambda < k+1. \end{cases} \tag{4.65}$$

(b) If $k' = k-1$ then:

$$q(k-1|k) = \frac{k}{\lambda} \longrightarrow h(k-1|k) = \begin{cases} \frac{k}{\lambda}, & \lambda \geq k, \\ 1, & \lambda < k. \end{cases} \tag{4.66}$$

(Note, in particular, that $h(-1|0) = 0$. So, the value -1 can be proposed but will never be accepted). The Metropolis et algorithm works in the following way:
– Chose randomly with probability $1/2$, $k' = k+1$ or $k' = k-1$.
– If $k' = k+1$, then if $k' \leq \lambda$ accept k'; if $k' > \lambda$ accept k' with probability $\frac{\lambda}{k+1}$.
– $k' = k-1$, then if $k \geq \lambda$ accept k'; if $k < \lambda$ accept k' with probability $\frac{k}{\lambda}$.

Here comes a possible version of the program:

```
integer function iran_poisson_mc(lambda)
implicit double precision(a-h,o-z)
double precision lambda
integer :: k=0
if (ran_u().gt.0.5d0) then
 k1=k+1
 if(k1.le.lambda) then
 k=k1
 iran_poisson_mc=k
 return
 elseif (ran_u().lt.lambda/k1) then
 k=k1
 iran_poisson_mc=k
 return
 endif
else
 k1=k-1
 if(k.ge.lambda) then
 k=k1
 iran_poisson_mc=k
 return
 elseif (ran_u().lt.k/lambda) then
 k=k1
 iran_poisson_mc=k
 return
 endif
endif
iran_poisson_mc=k
end function iran_poisson_mc
```

The average acceptance probability could be computed in principle, using the discrete version of (4.44) with sums replacing integrals, but is is simpler to compute it numerically using a simple modification of the previous program. A numerical fit concludes that $\epsilon \approx 1 - 0.4\lambda^{-1/2}$ is a good fit to the numerical data. Here, we do not have any parameters than can be tuned, buy these could be introduced, for instance, by setting a proposal distribution $g(k'|k) = 1/(2L + 1)$ if $|k' - k| \leq L$. The integer number L could then be used to reduce the correlation time. We leave this as an exercise to the reader.

4.5 Multidimensional distributions

The two previous applications have served as an introduction to the Metropolis *et al.* algorithm. However, its real power lies in the sampling of high-dimensional distributions where the random variable has N components, with N large. We will use the notation $x = (s_1, s_2, \dots, s_N)$ to denote the N dimensional random variable

$\hat{\mathbf{x}}$ we need to sample. Let us begin by an explicit pdf for $N = 4$:

$$\begin{aligned} f_{\hat{\mathbf{x}}}(s_1, s_2, s_3, s_4) &= A \exp\left(-s_1^4 - s_2^4 - s_3^4 - s_4^4 + s_1 s_2 + s_2 s_3 + s_3 s_4 + s_4 s_1\right) \\ &\equiv A\mathrm{e}^{d(x)}, \end{aligned} \tag{4.67}$$

where A is an irrelevant normalization factor and we have defined $d(x) = -s_1^4 - s_2^4 - s_3^4 - s_4^4 + s_1 s_2 + s_2 s_3 + s_3 s_4 + s_4 s_1$. The dynamical method proposes a new value $x' = (s_1', s_2', s_3', s_4')$ drawn from a distribution $g(x'|x) = g(x'|x)$ and accepts it with probability $h(x'|x)$. A simple extension of the previous examples suggests the proposal:

$$\begin{aligned} s_1' &= s_1 + (2u_1 - 1)\Delta, \\ s_2' &= s_2 + (2u_2 - 1)\Delta, \\ s_3' &= s_3 + (2u_3 - 1)\Delta, \\ s_4' &= s_4 + (2u_4 - 1)\Delta, \end{aligned} \tag{4.68}$$

being u_1, u_2, u_3, u_4 independent $\hat{\mathbf{U}}(0,1)$ variables. This means that each proposal s_i' is drawn from a uniform distribution in $(s_i - \Delta, s_i + \Delta)$. Note that this proposal is symmetrical as $g(x'|x) = g(x|x')$. The acceptance probability can be chosen as $h(x|x') = \min(1, q(x|y)$ being:

$$q(x'|x) = \frac{f_{\hat{\mathbf{x}}}(x')}{f_{\hat{\mathbf{x}}}(x)} = \mathrm{e}^{d(x')-d(x)} \tag{4.69}$$

Hence $h(x'|x) = 1$ if $d(x') > d(x)$, and $h(x'|x) = \mathrm{e}^{d(x')-d(x)}$, otherwise. A routine to implement the generation of this 4-dimensional variable $\hat{\mathbf{x}}$ could be:

```
subroutine ran_f4(s1,s2,s3,s4,delta)
q1=s1+delta*(2*ran_u()-1)
q2=s2+delta*(2*ran_u()-1)
q3=s3+delta*(2*ran_u()-1)
q4=s4+delta*(2*ran_u()-1)
d1=-s1**4-s2**4-s3**4-s4**4+s1*s2+s2*s3+s3*s4+s4*s1
d2=-q1**4-q2**4-q3**4-q4**4+q1*q2+q2*q3+q3*q4+q4*q1
if(d2.lt.d1) then
 if (ran_u().gt.exp(d2-d1)) return
endif
s1=q1
s2=q2
s3=q3
s4=q4
return
end
```

A `call ran_f4(s1,s2,s3,s4,delta)` returns in (s_1, s_2, s_3, s_4) a value of the 4-dimensional random variable $\hat{\mathbf{x}}$. Obviously, it is possible to replace

```
if(d2.lt.d1) then
 if (ran_u().gt.exp(d2-d1)) return
endif
```

by the single line

```
if (ran_u().gt.exp(d2-d1)) return
```

as the condition inside the `if` will never be satisfied for $d_1 < d_2$. However, this replacement implies to compute every time an exponential and generate a random number whether we need it or not, and the three-lines code is more efficient, albeit not so clear. The average acceptance probability depends on Δ. It would be difficult to compute it analytically, but the numerical results tell us that the average acceptance decays exponentially with the parameter Δ. This simply means again that the farther we allow the proposal to be with respect to the actual value of the variable, the smaller the acceptance error and, hence, the more often the routine will return exactly the same value for the random number.

It should be clear now the great advantage of this algorithm: it can be easily generalized to the case in which the dimensionality N of the random variable $\hat{\mathbf{x}}$, or the actual number of variables $(s_1, s_2, \dots, s_N)$ is large. And by large, we mean any number that can be accommodated in the computer memory. Let us be more precise and consider the pdf:

$$f_{\hat{\mathbf{x}}}(s_1, \dots, s_N) = A \exp\left(\sum_{i=1}^{N}[-s_i^4 + s_i s_{i+1}]\right). \tag{4.70}$$

We have used the convention, named "periodic boundary conditions" that whenever it appears, s_{N+1} must be replaced by s_1, and (this will be needed later) s_0 must be replaced by s_N, as if the variables were lying in a ring. It should be clear now the generalization of the Metropolis *et al.* algorithm, as shown in the following program.

```
subroutine ran_fn(s,n,delta)
implicit double precision (a-h,o-z)
dimension s(n),q(n)
common /acc/ d
do i=1,n
 q(i)=s(i)+delta*(2*ran_u()-1)
enddo
dn=0.0d0
do i=1,n
 i1=i+1
 if (i.eq.n) i1=1
dn=dn-q(i)**4+q(i)*q(i1)
enddo
if(dn.lt.d) then
 if (ran_u().gt.exp(dn-d)) return
endif
s=q
d=dn
return
end
```

Notice that we keep the value of the function $d(x)$ so we only need to compute its new value.

If we fix now the value of Δ, say $\Delta = 0.1$, the average acceptance decays with N. For instance for $N = 10$, it is $\epsilon \approx 0.852$, whereas for $N = 10^3$ it is $\epsilon \approx 4.8 \times 10^{-2}$ and for $N = 10^4$ it is $\epsilon \approx \times 10^{-6}$. It is clear then, that the average probability can become very small for N large. We could compensate this by decreasing Δ with N such that the average acceptance probability remains approximately constant.

It is true that a low acceptance probability is not, per se, a fundamental problem. What we care about is the correlation time τ_G, as the error increases with τ_G according to (4.24). But we believe, and we will sustain this belief later, that a small acceptance probability implies a large correlation time, as may times the values of the random variable will be identical, as the new proposals get rejected once and again.

The standard way of having a constant average acceptance with N consists in modifying the proposal $g(x'|x)$ in such a way that only one of the N variables $(s_1, \dots, s_N)$ is proposed to change. Formally, we use a proposal pdf:

$$g(x'|x) = \sum_{i=1}^{N} \frac{1}{N} g_i(s_i'|s_i) \prod_{j\neq i} \delta(s_j' - s_j). \tag{4.71}$$

This means that we select randomly one of the s_i variables, $i = 1, \dots, N$ (each one with probability $1/N$), propose a value s_i' from a probability distribution $g_i(s_i'|s_i)$ and keep all other variables with $j \neq i$ unchanged. For the single-variable proposal probability we take the same as before:

$$g_i(s_i'|s_i) = \frac{1}{2\Delta}, \quad \text{if } |s_i' - s_i| < \Delta, \tag{4.72}$$

or s_i' chosen randomly and uniformly from the interval $(s_i - \Delta, s_i + \Delta)$. It is clear that this proposal satisfies the symmetry condition $g(x'|x) = g(x|x')$ and hence the function $q(x'|x)$ is given by the ratio $f_{\hat{\mathbf{x}}}(x')/f_{\hat{\mathbf{x}}}(x)$. What is more interesting is that the calculation of this ratio implies only a few variables since most of them are unchanged and simplify in the numerator and denominator. Specifically, we have:

$$\begin{aligned} q(x'|x) &= \frac{f_{\hat{\mathbf{x}}}(s_1, \dots, s_i', \dots, s_N)}{f_{\hat{\mathbf{x}}}(s_1, \dots, s_i, \dots, s_N)} \\ &= \exp\left[-s_i'^4 + s_i^4 + (s_{i-1} + s_{i+1})(s_i' - s_i)\right] \end{aligned} \tag{4.73}$$

(recall our periodic boundary conditions convention: $s_0 \equiv s_N$ and $s_{N+1} \equiv s_1$).

Using the choice $h(x'|x) = \min(1, q(x'|x))$ for the acceptance probability, the algorithm would be:

```
subroutine ran_fn_1(s,n,delta)
implicit double precision (a-h,o-z)
dimension s(n)
do k=1,n
 i=i_ran()
 q=s(i)+delta*(2*ran_u()-1)
 ip=i+1
 if (i.eq.N) ip=1
 im=i-1
 if (i.eq.1) im=N
 d=-q**4+s(i)**4+(s(im)+s(ip))*(q-s(i))
 if(d.lt.0) then
 if (ran_u().gt.exp(d)) cycle
 endif
 s(i)=q
enddo
return
end
```

When we take, for example, $\Delta = 2.0$, the average acceptance probability is $\epsilon \approx 0.43$ independently on the value of N, whatever large this would be.

Note that in the previous program, we have repeated the basic proposal/acceptance step N times using the loop `do k=1,n`. This makes sense, as in every single proposal/acceptance step we only modify at most one of the N variables and the other $N-1$ remain unchanged. This repetition of N times the basic step is what is called *one Monte Carlo step* or 1 MCS. A new value of the N-dimensional random variable $\hat{\mathbf{x}}$ is only returned after each Monte Carlo step.

The choice (4.71), and, as it becomes clear in the previous program listing, implies a *random update*. At each time, the variable to be updated is chosen randomly amongst the set of N variables. It is possible to use the so-called *sequential update* in which the variables are chosen one after the other: first s_1, next s_2, then s_3, etc. From the practical point of view it simply means to replace the line `i=iran()` by `i=k`, or remove that line and change the loop variable to `do i=1,n`. There are several advantages to this procedure. First, it is faster since it avoids calling the integer random number routine. Second, it makes sure that every Monte Carlo step, each and every variable has been selected one for updating. Note that in random update, the probability that one variable has not been chosen after 1 MCS (N individual proposals) is $(1-1/N)^N$ which tends to $e^{-1} = 0.368$ for large N. The most notorious implication is that the Markov chain is not homogeneous as the transition probability $f_{\hat{\mathbf{x}}_n}(x'|x)$ depends now on n. In a sequential update at the 1st trial we chose variable s_1, at second, s_2 and so on, such at at trial n we chose variable s_{i_n} with $i_n = n - \left[\frac{n-1}{N}\right] N$, and the transition probability $f_{\hat{\mathbf{x}}_n}(x'|x) = f_{i_n}(x'|x)$, depends on the number i_n only. As before, once we have chosen variable s_{i_n} for updating, the proposal and acceptance probabilities are given by the corresponding expressions $g_{i_n}(x|x')$ and acceptance probabilities $h_{i_n}(x'|x)$ that now depend on i_n. The evolution of the pdf for this non-homogeneous process is:

$$\begin{aligned} f_{\hat{\mathbf{x}}_{n+1}}(x) &= \int f_{i_n}(x|y) f_{\hat{\mathbf{x}}_n}(y)\,dy \\ &= \int h_{i_n}(x|y) g_{i_n}(x|y) f_{\hat{\mathbf{x}}_n}(y)\,dy \\ &\quad + f_{\hat{\mathbf{x}}_n}(x)\left[1 - \int dz\, h_{i_n}(z|x) g_{i_n}(z|x)\right]. \end{aligned} \tag{4.74}$$

As we chose $g_{i_n}(x|x')$ and $h_{i_n}(x'|x)$ to satisfy the detailed balance condition $g_{i_n}(x|x')h_{i_n}(x'|x)f_{\hat{\mathbf{x}}}(x) = g_{i_n}(x'|x)h_{i_n}(x|x')f_{\hat{\mathbf{x}}}(x')$ for each value of i_n, then it is easy to prove that the distribution $f_{\hat{\mathbf{x}}}(x)$ is still a stationary distribution for (4.74). We need only to care about ergodicity and make sure that the sequential update allows, in principle, any possible values for the variables to be reached to make sure that $f_{\hat{\mathbf{x}}}(x)$ will be asymptotically sampled by the non-homogeneous Markov chain.

4.6 Heat-bath method

At variance with the previous methods, heat-bath is the first one that requires the variable to sample to be N-dimensional $\hat{\mathbf{x}} = (\hat{\mathbf{s}}_1, \dots, \hat{\mathbf{s}}_N)$ with $N > 1$. The idea is to solve the detailed balance condition (4.38) by (i) proposing transitions $x \to x'$ in which only one variable changes, $s_i \to s_i'$, and (ii) taking an acceptance probability equal to 1, $h(x'|x) = 1$. The detailed balance condition is reduced to:

$$g(x'|x) f_{\hat{\mathbf{x}}}(x) = g(x|x') f_{\hat{\mathbf{x}}}(x') \tag{4.75}$$

with $x = (s_1, \dots, s_i, \dots, s_N)$ and $x' = (s_1, \dots, s_i', \dots, s_N)$. This equation is satisfied if we take:

$$g(x'|x) = f(s_i'|s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_N). \tag{4.76}$$

This is easily checked by replacing

$$\begin{aligned} g(x'|x) &= f(s_i'|s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_N) \\ &= \frac{f_{\hat{\mathbf{x}}}(s_1, \dots, s_{i-1}, s_i', s_{i+1}, \dots, s_N)}{f(s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_N)} \end{aligned} \tag{4.77}$$

and

$$\begin{aligned} g(x|x') &= f(s_i|s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_N) \\ &= \frac{f_{\hat{\mathbf{x}}}(s_1, \dots, s_{i-1}, s_i, s_{i+1}, \dots, s_N)}{f(s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_N)} \end{aligned} \tag{4.78}$$

in (4.75).

In words, the heat-bath algorithm proceeds as follows:
(i) Chose a variable s_i. This step can be done either randomly ore sequentially.
(ii) Sample a new value s_i' from a conditional distribution in which all other variables $(s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_N)$ are fixed. The proposed value is always accepted. We will see examples of the use of this algorithm when we discuss in next chapter 5 some applications to systems of interest in statistical mechanics.

4.7 Tuning the algorithms

Parameter tuning. In a Monte Carlo simulation, it is very clear what we mean by the optimal algorithm: the one that gives the less error using the same computer time. If we remember that the error of the estimator is $\sigma[\hat{\mu}_M]$ as given by (4.24), and besides increasing the number of sample M (which obviously increases the computer time), one should try to reduce the correlation time τ_G (without increasing the computer time). For instance, when choosing the proposal probability (4.57) for the Gaussian distribution (but similar arguments apply to other cases) Δ is a parameter that can be varied without variation of the computer time. It is clear that τ_G is a function of Δ. If Δ is very small, the proposal x' is very close to the actual value x and will be accepted very often. A series of numbers G_k in which G_{k+1} is very close to G_k has a large correlation time. Similarly, when Δ is very large, the acceptance probability is very small and most values are rejected. As a consequence, in many occasions it is $G_{k+1} = G_k$ and, again, the correlation time is very large. Is is expected, then, that there exists an optimal value of Δ which yields the minimum correlation time τ_G. Unfortunately it is usually not so easy to determine with precision the optimal value Δ. Analytically, the problem is very difficult. To know τ_G, according to (4.19) and (4.23) we need to know the correlation function $\langle G_k G_{k'} \rangle$, an average in the stationary distribution. This is difficult as the iteration equation (4.42) can rarely be solved explicitly. It is possible to obtain an expression for the one-step correlation using

$$\begin{aligned}\langle G_{n+1} G_n \rangle &= \int dx\, dy\, G(x) G(y) f_{\hat{\mathbf{x}}_{n+1},\hat{\mathbf{x}}_n}(x,y) \\ &= \int dx\, dy\, G(x) G(y) f(x|y) f_{\hat{\mathbf{x}}_n}(y). \qquad (4.79)\end{aligned}$$

Replacing $f(x|y)$ from (4.36) and $f_{\hat{\mathbf{x}}_n}(y)$ for the stationary distribution $f_{\hat{\mathbf{x}}}(y)$, one obtains after some straightforward algebra that the one-time normalized correlation function is:

$$\begin{aligned}\rho_G(1) &= \frac{\langle G_{n+1} G_n \rangle - \langle G_k \rangle^2}{\sigma^2[G]} \\ &= 1 - \frac{1}{2\sigma^2[G]} \int dx\, dy\, h(x|y) g(x|y) [G(x) - G(y)]^2 f_{\hat{\mathbf{x}}}(y). \qquad (4.80)\end{aligned}$$

To obtain from here the correlation time we make the approximation that the correlation function decays exponentially $\rho_G(i) \approx [\rho_G(1)]^i$ and use (4.26). Although this approximation might not need necessarily accurate, it does give us an estimate for the correlation time τ_G.

For the Metropolis *et al.* algorithm applied to the Gaussian distribution, it is possible to perform analytically the integrals (4.80) to obtain after a tedious calculation that for $G(x) = x$ the one-time correlation function is:

$$\rho_x(1) = 1 - \frac{\Delta^2}{6}\left[1 - \operatorname{erf}\left(\frac{\Delta}{2\sqrt{2}}\right)\right] + \frac{16}{3\Delta\sqrt{2\pi}}\left[\left(\frac{\Delta^2}{8} + 1\right) e^{-\Delta^2/8} - 1\right] \quad (4.81)$$

where we can check that it has a minimum value at $\Delta \approx 3.70$, which is hence the optimal value for this parameter.

In general, it will not be possible to perform the integrals (4.80), but one can always compute the one-time correlation function $\rho_G(1)$ from the numerical algorithm and estimate numerically the correlation time τ_G using (4.26). This is relatively easy to do. Alternatively, one could compute numerically the whole correlation function $\rho_G(k)$ and derive the correlation time from there. We have left for appendix 14 the explanation of an efficient algorithm to compute the correlation time of a series.

If all the above seems too complicated (although it should not really be), there is another strategy to determine the optimal value of the tunable parameters. We have already commented on the fact that very large or very small acceptance probabilities lead to large correlation times. The simple answer then is to chose parameters such that the average acceptance probability is not very large, nor very small. Not very large, nor very small for a number ϵ which is bounded to be between 0 and 1 means ... $\epsilon = 1/2$. This is a heuristic rule, with no much justification, but to chose parameters using this criterion is usually better than not tuning parameters at all. For instance, at the optimal value $\Delta = 3.70$ for the Gaussian distribution (the minimum of the one-time correlation function), the acceptance probability (as given by (4.61)) is $\epsilon = 0.418$, not equal to $1/2$, but not too far from it either.

How often? The computer time spent in the calculation of the estimator $\hat{\mu}_M[G]$ to the integral of the function $G(x)$ can be divided in the time necessary, say t_2, to generate one value x_k of the random variable and the time, say t_1, it takes to compute the function $G(x_k)$ and make the necessary sums, products and sums of squares to compute the estimator, the error, the correlation function, and so on. If the correlation time τ_G is small (of the order of 1) then there is little correlation between all values of $G(x_k)$ and including them all in the averages makes perfect sense. Imagine, however (and, alas!, this occurs more often that one desires), that the correlation time τ_G is large. Then there is a large correlation between the different values of $G(x_k)$ and it might not be useful to include all of them in the calculation of the averages. This means to let pass a number of proposal/acceptance steps, say K, between measurements. In this way, the correlation time of the new, decimated, series of values $G(x_k)$ has a correlation time τ_G/K. Is there an optimal value for K? The answer is yes, and a simple calculation of its optimal value can save us a lot of computer time in complicated problems, where the calculation of $G(x_k)$ is time consuming.

So, let us assume that we make M measurements and that there are K proposal/acceptance steps between measurements. The correlation time is τ_G/K, and the variance of the estimator using these values is:

$$\sigma^2[\hat{\mathbf{G}}] = \frac{\sigma^2[\hat{\mathbf{G}}]}{M}\left(2\frac{\tau_G}{K}+1\right) \tag{4.82}$$

and we would like to chose K to minimize this expression. The minimization has to be performed under the condition that the total computer time is constant. The total computer time is $T_2 = KMt_2$ (time needed to generate the KM steps) plus $T_1 = Mt_1$ (measurement time). The minimization of

$$\frac{1}{M}\left(2\frac{\tau_G}{K}+1\right) \tag{4.83}$$

with the constrain

$$Mt_1 + KMt_2 = t \text{ constant}, \tag{4.84}$$

yields

$$K = \sqrt{\frac{2\tau_G t_1}{t_2}}, \tag{4.85}$$

as the optimal value of K. The ratio of the time spent in measuring to the one spent in updating is:

$$\frac{T_1}{T_2} = \frac{Mt_1}{KMt_2} = \sqrt{\frac{t_1}{2\tau_G t_2}}. \tag{4.86}$$

This time does not seem to be an easy heuristic rule to replace this calculation. Sometimes it is used the criterion of choosing K such that the ratio of measuring to updating is $T_1/T_2 = 1$ or $K = t_1/t_2$, a criterion not sustained by this simple calculation.

Thermalization. We reach now a delicate point. As mentioned in section 1.8 the use by a dynamical methods of the adequate homogeneous Markov chain, and under the assumption of ergodicity, ensures that the values of the random variable $\hat{\mathbf{x}}$ from a pdf $f_{\hat{\mathbf{x}}}(x)$ are reached **asymptotically**. Namely,

$$\lim_{n\to\infty} f_{\hat{\mathbf{x}}_n}(x) = f_{\hat{\mathbf{x}}}(x). \tag{4.87}$$

According to this criterion, we should wait an "infinite" number of steps before reaching the stationary state and start the measurements.

We have already mentioned that, in practice "$n \to \infty$" means that we have to discard the first M_0 steps of the algorithm, the first M_0 values of the random variable $\hat{\mathbf{x}}$. The problem is to determine faithfully the value of M_0. We can affirm that underestimating the value of M_0 is the main source of uncontrolled mistakes in the field of Monte Carlo simulations and there are many the published papers which are wrong just because the authors did not wait long enough to reach the stationary state of the Markov chain.

The process of discarding the first M_0 steps is called **thermalization**. So, the key question is how long should we thermalize? How to get an estimate of M_0? Obviously, one thing to do is to vary M_0 and check that the results, without the unavoidable statistical errors, do not depend on M_0. One can be a little bit more precise and compute the time dependence of the averages. For that we need to compute the so-called lineal relaxation function ρ_G^{NL} defined as:

$$\rho_G^{\text{NL}}(n) = \frac{\langle G_n\rangle - \langle G\rangle_{\text{st}}}{\langle G_0\rangle - \langle G\rangle_{\text{st}}}. \tag{4.88}$$

Here

$$\langle G_n\rangle = \int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}_n}(x)G(x)\,dx \tag{4.89}$$

denotes an average using the different values that are generated at step n using a different value from the initial pdf $f_{\hat{\mathbf{x}}_0}(x)$ and $\langle G\rangle_{\text{st}}$ is the stationary value. The thermalization time is related to the time it takes $\rho_G^{\text{NL}}(n)$ to decay from $\rho_G^{\text{NL}}(n) = 1$ to its stationary value $\rho_G^{\text{NL}}(n) = 0$. This can be estimated similarly to the correlation function introducing the non-linear correlation time:

$$\tau_G^{\text{NL}} = \sum_{n=0}^{\infty} \rho_G^{\text{NL}}(n), \tag{4.90}$$

and, finally, taking M_0 as a number of times τ_G^{NL}, for example $M_0 \geq 10\tau_G^{\text{NL}}$. It is clear that the problem in this procedure is that the definition (4.88) depends on the stationary value $\langle G\rangle_{\text{st}}$ which can only be determined free of systematic errors if we are sure that we are in the stationary state. There are examples (mostly in the field of phase transitions, but not only there) in which the decay is very slow and it is difficult to determine whether or not the system has reached indeed the stationary state.

There is one check, however, that can help us in this issue. There is a simple relation

$$\langle q(y|x)\rangle_{\text{st}} = 1 \tag{4.91}$$

that holds in the stationary state. The average has to be performed with respect to the **proposed** values y (hence, distributed according to $g(y|x)$) over the allegedly stationary distribution of x. Namely,

$$\langle q(y|x)\rangle_{\text{st}} = \int dy \int dx f_{\hat{\mathbf{x}}}(x) g(y|x) q(y|x). \tag{4.92}$$

The proof is simple, replace $q(x|y)$ by its definition and manipulate:

$$\begin{aligned}
\langle q(y|x)\rangle_{\text{st}} &= \int dy \int dx f_{\hat{\mathbf{x}}}(x) g(y|x) q(y|x) \\
&= \int dy \int dx f_{\hat{\mathbf{x}}}(x) g(y|x) \frac{g(x|y) f_{\hat{\mathbf{x}}}(y)}{g(y|x) f_{\hat{\mathbf{x}}}(x)} \\
&= \int dy \left[\int dx g(x|y)\right] f_{\hat{\mathbf{x}}}(y) \\
&= \int dy f_{\hat{\mathbf{x}}}(y) = 1.
\end{aligned} \tag{4.93}$$

Therefore, if we are in the stationary state, we should verify (4.91). Unfortunately, this is a necessary condition, but not sufficient and one cat get values of $\langle q(x|y)\rangle_{\text{st}}$ close to 1 and still not be in the stationary state. In practice, one should get a value of $\langle q(x|y)\rangle_{\text{st}}$ really close to 1, say a zero followed by a good number of nines (at least three or four). Otherwise, you most probably have not yet reached the stationary state.

Exercises

1) Sample the N-dimensional distribution (4.30) using a simple rejection method with $(x_1, \ldots, x_N)$ independent and uniformly distributed in the interval $[-1, 1]$ and check that the average acceptance probability decays exponentially with N.
2) Prove that in a homogenous Markov chain, the correlation function $\rho(i, j)$ depends only on the difference $|i - j|$.
3) Use (4.44) with the proposal (4.57) and the Metropolis *et al.* choice $h(x|y) = \min(1, q(x|y))$, to reproduce expression (4.61). Check the correctness of this result by comparing with numerical simulations varying the parameter Δ. Compare (numerically) with the average acceptance probability ϵ obtained using Glauber choice $h(x|y) = \dfrac{q(x|y)}{1 + q(x|y)}$. Which method has the largest acceptance probability?
4) Continuing with the previous problem, compute numerically, both for Metropolis *et al.* and Glauber acceptance probabilities, the correlation time in the stationary state for the function $G(x) = x^2$ as a function of the parameter Δ and determine the optimal values of Δ. Conclude which choice is more efficient (do not forget to take into consideration the computer time needed by each algorithm).
5) Repeat the previous problem using the function $G(x) = \cos(x)$ and check that the numerical value obtained for the integral

$$\int_{-\infty}^{\infty} dx \cos(x) \exp(-\frac{x^2}{2})$$

using the Metropolis *et al.* algorithm for both choices of the acceptance probability $h(x|y)$ agrees within errors with the exact result $\sqrt{2\pi/e}$.
6) Compute as a function of λ the correlation time of the Metropolis *et al.* algorithm applied to the generation of the Poisson distribution of parameter λ. Take $\lambda = 0.5, 1.0, 2.0, 5.0, 10.0, 100.0$
7) Program the Metropolis algorithm for the generation of the Poisson distribution using a proposal $g(k'|k)$ uniformly distributed in the interval $(k - L, k + L)$. Compute the correlation time τ_k associated to the function $G(k) = k$ and find the value of L that makes τ_G minimum and discuss its dependence on the parameter λ of the Poisson distribution.
8) Consider a 3-dimensional discrete random variable $\hat{\mathbf{x}}$ that can take values in $\Omega = \{-1, 1\}^3$. In other words, the possible values are: $x_1 = (1, 1, 1)$, $x_2 = (1, 1, -1)$, $x_3 = (1, -1, 1)$, $x_4 = (1, -1, -1)$, $x_5 = (-1, 1, 1)$, $x_6 = (-1, 1, -1)$, $x_7 = (-1, -1, 1)$, $x_8 = (-1, -1, -1)$. Writing $x_i = (s_1^i, s_2^i, s_3^i)$ we assign to each possible value x_i a probability p_i:

$$p_i = C \exp[K(s_1^i s_2^i + s_2^i s_3^i + s_3^i s_1^i)]$$

where K is a given number, and C the normalization constant. Write programs that implement Metropolis *et al.* algorithm with (i) Metropolis *et al.* (ii) Glauber and (iii) van Beijeren and Schulman choices for the acceptance probability to generate values of $\hat{\mathbf{x}}$ distributed according to these probabilities. Use proposal probabilities (i) in which only one of the s_i variables is proposed for change (which is the only possible proposal?) and (ii) in which the proposed value is (s'_1, s'_2, s'_3) is randomly chosen from the 8 possible values.

9) Given the functions $M(x) = (s_1 + s_2 + s_3)/3$ and $E(x) = (s_1 s_2 + s_2 s_3 + s_3 s_1)/3$ defined on space Ω of the previous problem, compute the linear and non-linear correlation times of the Metropolis *et al.* algorithm. Use this result to compute $\langle M\rangle$, $\langle M^2\rangle$, $\langle E\rangle$, $\langle E^2\rangle$ as a function of K (take $K = 0.25, 0.5, \dots, 4.0$). Compare with the exact results obtained computing the averages summing over the 8 possible values of the variable $\hat{\mathbf{x}}$.
10) Repeat the two previous exercises for the space $\Omega = \{-1, 1\}^{1000}$. Which is the only proposal that gives a non-vanishing acceptance probability now?
11) Consider the Metropolis *et al.* method applied to the distribution (4.70). Use a proposal $g(y|x)$ in which **all** variables s_i, $i = 1, \dots, N$ are proposed a new value in the interval $(s_i - \Delta, s_i + \Delta)$. Determine the dependence of Δ with the number of variables N such that the acceptance probability is constant and close to 0.5. Using this value compute the correlation time as a function of N. Compare its value with the one obtained by the method in which **only one** variable is chosen for updating at a given step and Δ does not depend on N. Which method is more efficient?
12) Use the Glauber acceptance probability (4.55) to prove that, under the assumption of a Gaussian distribution for the proposed changes of energy $\Delta\mathcal{H}$, the average acceptance probability decays as $h(y|x) \to \frac{e^{-\frac{\beta\langle\Delta\mathcal{H}\rangle}{4}}}{\sqrt{\beta\langle\Delta\mathcal{H}\rangle}}$, still a exponential dependence with the average change of energy proposal $\Delta\mathcal{H}$.
13) Prove that K given by (4.85) is the optimal choice for the number of updates between measurements in order to minimize the computer time.
14) For the rejection with repetition method explained in section 4.1 compute analytically the correlation function $C_{\text{st}}(n) = \frac{\langle\hat{\mathbf{x}}_n\hat{\mathbf{x}}_0\rangle - \langle\hat{\mathbf{x}}_n\rangle\langle\hat{\mathbf{x}}_0\rangle}{\sigma^2[\hat{\mathbf{x}}_0]}$, in the steady state, as well as the non-linear correlation function

$$C_{\text{nl}}(n) = \frac{\langle\hat{\mathbf{x}}_n\rangle - \langle\hat{\mathbf{x}}_\infty\rangle}{\langle\hat{\mathbf{x}}_0\rangle - \langle\hat{\mathbf{x}}_\infty\rangle}$$