Four lectures on computational statistical physics

Werner Krauth[1].

Laboratoire de Physique Statistique
CNRS-Ecole Normale Supérieure
24 rue Lhomond
75231 Paris Cedex 05

[1]Also available at arXiv:0901.2496. Lectures given at the 2008 Les Houches Summer School "Exact methods in low-dimensional physics and quantum computing".
Parts 1.1 and 3.1 - 3.3 also presented at the workshop "Numerical Approaches to Quantum Many-Body Systems", Institute for Pure and Applied Mathematics, UCLA, January 2009.

Preface

In my lectures at the Les Houches Summer School 2008, I discussed central concepts of computational statistical physics, which I felt would be accessible to the very cross-cultural audience at the school.

I started with a discussion of sampling, which lies at the heart of the Monte Carlo approach. I specially emphasized the concept of perfect sampling, which offers a synthesis of the traditional direct and Markov-chain sampling approaches. The second lecture concerned classical hard-sphere systems, which illuminate the foundations of statistical mechanics, but also illustrate the curious difficulties that beset even the most recent simulations. I then moved on, in the third lecture, to quantum Monte Carlo methods, that underly much of the modern work in bosonic systems. Quantum Monte Carlo is an intricate subject. Yet one can discuss it in simplified settings (the single-particle free propagator, ideal bosons) and write direct-sampling algorithms for the two cases in two or three dozen lines of code only. These idealized algorithms illustrate many of the crucial ideas in the field. The fourth lecture attempted to illustrate aspects of the unity of physics as realized in the Ising model simulations of recent years.

More details on what I discussed in Les Houches, and wrote up (and somewhat rearranged) here, can be found in my book, "Statistical Mechanics: Algorithms and Computations" (SMAC), as well as in recent papers. Computer programs are available for download and perusal at the book's web site www.smac.lps.ens.fr.

Contents

1
Sampling

1.1 Direct sampling, sample transformation

As an illustration of what is meant by sampling, and how it relates to integration, we consider the Gaussian integral:

$$\int_{-\infty}^{\infty} \frac{\mathrm{d}x}{\sqrt{2\pi}} \exp\left(-x^2/2\right) = 1. \tag{1.1}$$

This integral can be computed by taking its square:

$$\left[\int_{-\infty}^{\infty} \frac{\mathrm{d}x}{\sqrt{2\pi}} \exp\left(-x^2/2\right)\right]^2 = \int_{-\infty}^{\infty} \frac{\mathrm{d}x}{\sqrt{2\pi}} \mathrm{e}^{-x^2/2} \int_{-\infty}^{\infty} \frac{\mathrm{d}y}{\sqrt{2\pi}} \mathrm{e}^{-y^2/2} \tag{1.2}$$

$$\ldots = \int_{-\infty}^{\infty} \frac{\mathrm{d}x\mathrm{d}y}{2\pi} \exp\left[-(x^2+y^2)/2\right], \tag{1.3}$$

and then switching to polar coordinates ($\mathrm{d}x\mathrm{d}y = r\mathrm{d}r\mathrm{d}\phi$),

$$\ldots = \int_0^{2\pi} \frac{\mathrm{d}\phi}{2\pi} \int_0^{\infty} r\mathrm{d}r \ \exp\left(-r^2/2\right), \tag{1.4}$$

as well as performing the substitutions $r^2/2 = \Upsilon$ ($r\mathrm{d}r = \mathrm{d}\Upsilon$) and $\exp(-\Upsilon) = \Psi$

$$\ldots = \int_0^{2\pi} \frac{\mathrm{d}\phi}{2\pi} \int_0^{\infty} \mathrm{d}\Upsilon \ \exp(-\Upsilon) \tag{1.5}$$

$$\ldots = \int_0^{2\pi} \frac{\mathrm{d}\phi}{2\pi} \int_0^{1} \mathrm{d}\Psi \ = 1. \tag{1.6}$$

In our context, it is less important that we can do the integrals in eqn (1.6) analytically, than that ϕ and Ψ can be sampled as uniform random variables in the interval $[0, 2\pi]$ (for ϕ) and in $[0, 1]$ (for Ψ). Samples $\phi = \texttt{ran}(0, 2\pi)$ and $\Psi = \texttt{ran}(0, 1)$, are readily obtained from the random number generator $\texttt{ran}(a, b)$ lingering on any computer. We can plug these random numbers into the substitution formulas which took us from eqn (1.2) to eqn (1.6), and that take us now from two uniform random numbers ϕ and Ψ to Gaussian random numbers x and y. We may thus apply the integral transformations in the above equation to the samples, in other words perform a "sample transformation". This is a practical procedure for generating Gaussian random numbers from uniform random numbers, and we best discuss it as what it is, namely an algorithm.

procedure gauss
$\phi \leftarrow \texttt{ran}\,(0, 2\pi)$
$\Psi \leftarrow \texttt{ran}\,(0, 1)$
$\Upsilon \leftarrow -\texttt{log}\ \Psi$
$r \leftarrow \sqrt{2\Upsilon}$
$x \leftarrow r\texttt{cos}\ \phi$
$y \leftarrow r\texttt{sin}\ \phi$
output $\{x, y\}$

Algorithm 1 gauss. SMAC pseudocode program for transforming two uniform random numbers ϕ, Ψ into two independent uniform Gaussian random numbers x, y. Except for typography, SMAC pseudocode is close to the Python programming language.

SMAC pseudocode can be implemented in many computer languages [1]. Of particular interest is the computer language Python, which resembles pseudocode, but is executable on a computer exactly as written. We continue in these lectures using and showing Python code, as in algorithm 1.1.

Algorithm 1.1 gausstest.py

```
from random import uniform as ran, gauss
from math import sin, cos, sqrt, log, pi
def gausstest(sigma):
  phi = ran(0,2*pi)
  Upsilon = -log(ran(0.,1.))
  r = sigma*sqrt(2*Upsilon)
  x = r*cos(phi)
  y = r*sin(phi)
  return x,y
print gausstest(1.)
```

Direct-sampling algorithms exist for arbitrary one-dimensional distributions (see SMAC Sect. 1.2.4). Furthermore, arbitrary discrete distributions $\{\pi_1, \ldots, \pi_K\}$ can be directly sampled, after an initial effort of about K operations with $\sim \log_2 K$ operation by "Tower sampling" (see SMAC Sect. 1.2.3). This means that sampling a distribution made up of, say, one billion terms takes only about 30 steps. We will use this fact in Section 3.3, for the direct sampling algorithm for ideal bosons. Many trivial multi-dimensional distribution (as for example non-interacting particles) can also be sampled.

Direct-sampling algorithms also solve much less trivial problems as for example the free path integral, ideal bosons, and the two-dimensional Ising model, in fact, many problems which possess an exact analytic solution. These direct-sampling algorithms are often the race-car engines inside general-purpose Markov-chain algorithms for complicated interacting problems.

Let us discuss the computation of integrals derived from the sampled ones $Z = \int \mathrm{d}x\ \pi(x)$ (in our example, the distribution $\pi(x)$ is the Gaussian)

[1] The SMAC web page www.smac.lps.ens.fr provides programs in language ranging from Python, Fortran, C, and Mathematica to TI basic, the language of some pocket calculators.

$$\frac{\int \mathrm{d}x\ \mathcal{O}(x)\pi(x)}{\int \mathrm{d}x\ \pi(x)} \simeq \frac{1}{N}\sum_{i=1}^{N}\mathcal{O}(x_i),$$

where the points x_i are sampled from the distribution $\pi(x)$. This approach, as shown, allows to compute mean values of observables for a given distribution π. One can also bias the distribution function π using Markov-chain approaches. This gives access to a large class of very non-trivial distributions.

1.2 Markov-chain sampling

Before taking up the discussion of elaborate systems (hard spheres, bosons, spin glasses), we first concentrate on a single particle in a finite one-dimensional lattice $k = 1, \ldots, N$ (see Fig. 1.1). This case is even simpler than the aforementioned Gaussian, because the space is discrete rather than continuous, and because the site-occupation probabilities $\{\pi_1, \ldots, \pi_N\}$ are all the same

$$\pi_k = \frac{1}{N} \quad \forall k.$$

We can sample this trivial distribution by picking k as a random integer between 1 and N. Let us nevertheless study a Markov-chain algorithm, whose diffusive dynamics converges towards the probability distribution of eqn (1.2). For concreteness, we consider the algorithm where at all integer times $t \in]-\infty, \infty[$, the particle hops with probability $\frac{1}{3}$ from one site to each of its neighbors:

$$p_{k\to k+1} = p_{k\to k-1} = 1/3 \quad \text{(if possible)}. \tag{1.7}$$

The probabilities to remain on the site are 1/3 in the interior and, at the boundaries, $p_{1\to 1} = p_{N\to N} = 2/3$.

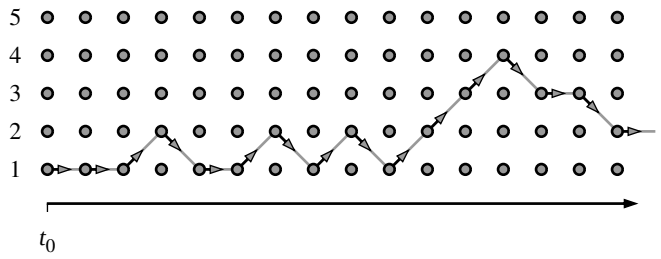

Fig. 1.1 Markov-chain algorithm on a five-site lattice. A single particle hops towards a site's neighbor with probability 1/3, and remains on the site with probability 1/3 (probability 2/3 on the boundary).

These transition probabilities satisfy the notorious detailed balance condition

$$\pi_k p_{k\to l} = \pi_l p_{l\to k} \tag{1.8}$$

for the constant probability distribution π_k. Together with the ergodicity of the algorithm this guarantees that in the infinite-time limit, the probability π_k to find the

particle at site k is indeed independent of k. With appropriate transition probabilities, the Metropolis algorithm would allow us to sample any generic distribution π_k or $\pi(x)$ in one or more dimensions (see SMAC Sect. 1.1.5).

Markov-chain methods are more general than direct approaches, but the price to pay is that they converge to the target distribution π only in the infinite-time limit. The nature of this convergence can be analyzed by the transfer matrix $T^{1,1}$ (the matrix of transition probabilities, in our case a 5×5 matrix):

$$T^{1,1} = \{p_{k\to l}\} = \frac{1}{3}\begin{pmatrix} 2&1&0&0&0\\ 1&1&1&0&0\\ 0&1&1&1&0\\ 0&0&1&1&1\\ 0&0&0&1&2 \end{pmatrix} \tag{1.9}$$

The eigenvalues of the transfer matrix $T^{1,1}$ are $(1, \frac{1}{2} \pm \sqrt{5}/6, \frac{1}{6} \pm \sqrt{5}/6)$. The largest eigenvalue, equal to one, expresses the conservation of the sum of all probabilities $\sum(\pi_1 + \cdots + \pi_5) = 1$. Its corresponding eigenvector is $|\bullet\circ\circ\circ\circ\rangle + \cdots + |\circ\circ\circ\circ\bullet\rangle$, by construction, because of the detailed-balance condition eqn (4.2). The second-largest eigenvalue, $\lambda_2^{1,1} = \frac{1}{2} + \sqrt{5}/6 = 0.8727$ governs the decay of correlation functions at large times. This is easily seen by computing the probability vector $\boldsymbol{\pi}(t_0 + \tau) = \{\pi_1(t), \ldots, \pi_5(t)\}$ which can be written in terms of the eigenvalues and eigenvectors

Let us suppose that the simulations always start[2] on site $k = 1$, and let us decompose this initial configuration onto the eigenvectors $\boldsymbol{\pi}(0) = \alpha_1\boldsymbol{\pi}_1^e + \cdots + \alpha_N\boldsymbol{\pi}_N^e$.

$$\boldsymbol{\pi}(t) = \left(T^{1,1}\right)^t \boldsymbol{\pi}(0) = \boldsymbol{\pi}_1^e + \alpha_2\lambda_2^t\boldsymbol{\pi}_2^e + \ldots$$

At large times, corrections to the equilibrium state vanish as λ_2^τ, so that site probabilities approach the equilibrium value as $\exp(\tau/\tau_{\rm corr})$ with a time constant

$$\tau_{\rm corr} = 1/|\log\lambda_2| \simeq 7.874 \tag{1.10}$$

(see SMAC Sect. 1.1.4, p. 19f).

We retain that the exponential convergence of a Monte Carlo algorithm is characterized by a scale, the convergence time $\tau_{\rm corr}$. We also retain that convergence takes place after a few $\tau_{\rm corr}$, in our example say $3 \times 7.87 \sim 25$ iterations (there is absolutely no need to simulate for an infinite time). If our Markov chains always start at $t = 0$ on the site 1, it is clear that for all times $t' < \infty$, the occupation probability $\pi_1(t')$ of site 1 is larger than, say, the probability $\pi_5(t')$ to be on site 5. This would make us believe that Markov chain simulations never completely decorrelate from the initial condition, and are always somehow less good than direct-sampling. This belief is wrong, as we shall discuss in the next section.

1.3 Perfect sampling

We need to simulate for no more than a few $\tau_{\rm corr}$, but we must not stop our calculation short of this time. This is the critical problem in many real-life situations, where we

[2] we should assume that the initial configuration is different from the stationary solution because otherwise all the coefficients $\alpha_2, \ldots, \alpha_N$ would be zero.

cannot compute the correlation time as reliably as in our five-site problem: $\tau_{\rm corr}$ may be much larger than we suspect, because the empirical approaches for determining correlation times may have failed (see SMAC Sect. 1.3.5). In contrast, the problem of the exponentially small tail for $\tau \gg \tau_{\rm corr}$ is totally irrelevant.

Let us take up again our five-site problem, with the goal of obtaining rigorous information about convergence from within the simulation. As illustrated in Fig. 1.2, the Markov-chain algorithm can be formulated in terms of time sequences of random maps: Instead of prescribing one move per time step, as we did in Fig. 1.1, we now sample moves independently for all sites k and each time t. At time t_0, for example, the particle should move straight from sites 1, 2 and 5 and down from sites 3 and 4, etc. Evidently, for a single particle, there is no difference between the two formulations, and detailed balance is satisfied either way. With the formulation in terms of random maps, we can verify that from time $t_0 + \tau_{\rm coup}$ on, all initial conditions generate the same output. This so-called "coupling" is of great interest because after the coupling time $\tau_{\rm coup}$, the influence of the initial condition has completely disappeared. In the rest of this lecture, we consider extended Monte Carlo simulations as the one in Fig. 1.2, with arrows drawn for each site and time.

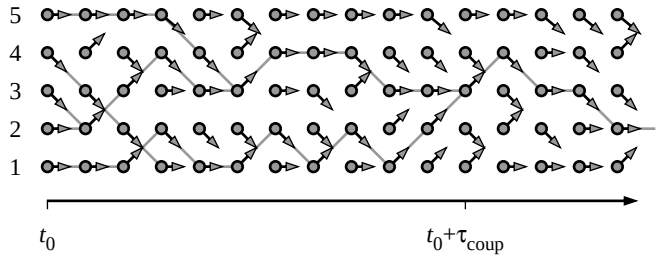

Fig. 1.2 Extended Monte Carlo simulation on $N = 5$ sites. In this example, the coupling time is $\tau_{\rm coup} = 11$.

The coupling time $\tau_{\rm coup}$ is a random variable ($\tau_{\rm coup} = 11$ in Fig. 1.2) whose distribution $\pi(\tau_{\rm coup})$ vanishes exponentially in the limit $\tau_{\rm coup} \to \infty$ because the random maps at different times are independent.

The extended Monte Carlo dynamics describes a physical system with from 1 to N particles, and transition probabilities, from eqn (1.7), as for example:

$$\begin{aligned} p^{\rm fw}(|\circ\circ\bullet\circ\bullet\rangle \to |\circ\bullet\circ\bullet\circ\rangle) &= 1/9 \\ p^{\rm fw}(|\circ\circ\bullet\circ\bullet\rangle \to |\circ\circ\bullet\circ\bullet\rangle) &= 2/9 \\ p^{\rm fw}(|\circ\circ\bullet\circ\bullet\rangle \to |\circ\circ\circ\bullet\circ\rangle) &= 1/9 \\ p^{\rm fw}(|\circ\circ\bullet\bullet\bullet\rangle \to |\circ\circ\circ\bullet\circ\rangle) &= 1/27, \end{aligned} \tag{1.11}$$

etc., We may analyze the convergence of this system through its transfer matrix $T^{\rm fw}$ (in our case, a 31×31 matrix, because of the 31 non-empty states on five sites):

$$T^{\mathrm{fw}} = \begin{pmatrix} T^{1,1} & T^{2,1} & \dots & \dots \\ 0 & T^{2,2} & T^{3,2} & \dots \\ 0 & 0 & T^{3,3} & \dots \\ \dots & & & \\ 0 & 0 & & T^{N,N} \end{pmatrix}$$

where the block $T^{k,l}$ connects states with k particles at time t to states with $l \leq k$ particles at time $t+1$. The matrix T^{fw} has an eigenvector with eigenvalue 1 which describes the equilibrium, and a second-largest eigenvalue which describes the approach to equilibrium, and yields the coupling time τ_{coup}. Because of the block-triangular structure of T^{fw}, the eigenvalues of $T^{1,1}$ constitute the single-particle sector of T^{fw}, including the largest eigenvalue $\lambda_1^{\mathrm{fw}} = 1$, with corresponding left eigenvector $|\bullet\circ\circ\circ\circ\rangle + \cdots + |\circ\circ\circ\circ\bullet\rangle$. The second-largest eigenvalue of T^{fw} belongs to the $(2,2)$ sector. It is given by $\lambda_2^{\mathrm{fw}} = 0.89720 > \lambda_2^{\mathrm{MC}}$, and it describes the behavior of the coupling probability $\pi(\tau_{\mathrm{coup}})$ for large times.

Markov chains shake off all memory of their initial conditions at time τ_{coup}, but this time changes from simulation to simulation: it is a random variable. Ensemble averages over (extended) simulations starting at t_0, and ending at $t_0 + \tau$ thus contain mixed averages over coupled and "not yet coupled" runs, and only the latter carry correlations with the initial condition. To reach pure ensemble averages over coupled simulations only, one has to start the simulations at time $t = -\infty$, and go up to $t = 0$. This procedure, termed "coupling from the past" (Propp and Wilson 1996), is familiar to theoretical physicists because in dynamical calculations, the initial condition is often put to $-\infty$. Let us now see how this trick can be put to use in practical calculations.

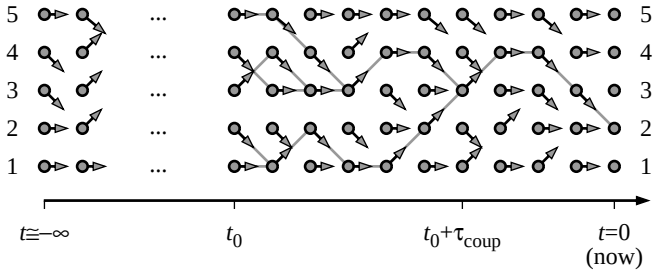

Fig. 1.3 Extended simulation on $N = 5$ sites. The outcome of this infinite simulation, from $t = -\infty$ through $t = 0$, is $k = 2$.

An extended simulation "from the past" is shown in Fig. 1.3. It has run for an infinite time so that all the eigenvalues of the transfer matrix but $\lambda_1 = 1$ have died away and only the equilibrium state has survived. The configuration at $t = 0$ is thus a perfect sample, and each value k ($k \in \{1, \dots, 5\}$ is equally likely, if we average over all extended Monte Carlo simulations. For the specific simulation (choice of arrows in Fig. 1.3), we know that the simulation must pass by one of the five points at time $t = t_0 = -10$). However, the Markov chain couples between $t = t_0$ and $t = 0$, so that we know that $k(t = 0) = 2$. If it did not couple at $t = t_0$, we would have to provide more information (draw more arrows, for $t = t_0 - 1, t_0 - 2, \dots$) until the chain couples.

Up to now, we have considered the forward transfer matrix, but there is also a backward matrix $T^{\rm bw}$, which describes the propagation of configurations from $t = 0$ back to $t = -1$, $t = -2$, etc. $A^{\rm bw}$ is similar to the matrix $A^{\rm fw} \sim A^{\rm bw}$ (the two matrices have identical eigenvalues), because the probability distribution to couple between time $t = -|t_0|$ and $t = 0$ equals the probability to couple between time $t = 0$ and $t = +|t_0|$. This implies that the distributions of coupling times in the forward and backward directions are identical.

We have also seen that the eigenvalue corresponding to the coupling time is larger than the one for the correlation time. For our one-dimensional diffusion problem, this can be proven exactly. More generally, this is because connected correlation functions decay as

$$\langle \mathcal{O}(0)\mathcal{O}(t)\rangle_c \sim \mathrm{e}^{-t/\tau_{\rm corr}}.$$

Only the non-coupled (extended) simulations contribute to this correlation function, and their proportion is equal to $\exp(-t/\tau_{\rm coup})$. We arrive at

$$\exp\left[-t\left(1/\tau_{\rm coup} - 1/\tau_{\rm corr}\right)\right] \sim \langle \mathcal{O}(0)\mathcal{O}(t)\rangle_c^{\text{non-coup}},$$

and $\tau_{\rm coup} > \tau_{\rm corr}$ because even the non-coupling correlation functions should decay with time.

Finally, we have computed in Fig. 1.2 and in Fig. 1.3 the coupling time by following all possible initial conditions. This approach is unpractical for more complicated problems, such as the Ising model on N sites with its 2^N configurations. Let us show in the five-site model how this problem can sometimes be overcome: It suffices to define a new Monte Carlo dynamics in which trajectories cannot cross, by updating, say, even lattice sites ($k = 2, 4$) at every other time $t_0, t_0 + 2, \ldots$ and odd lattice sites ($k = 1, 3, 5$) at times $t_0 + 1, t_0 + 3, \ldots$. The coupling of the trajectories starting at sites 1 and 5 obviously determines $\tau_{\rm coup}$.

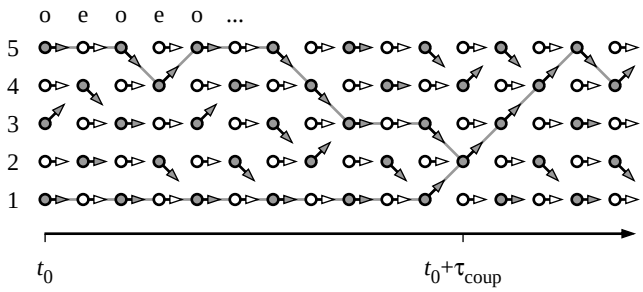

Fig. 1.4 Extended Monte Carlo simulation with odd and even times. Trajectories cannot cross, and the coupling of the two extremal configurations (with $k(t_0) = 1$ and $k(t_0 = 5)$) determines the coupling time.

In the Ising model, the ordering relation of Fig. 1.4 survives in the form of a "half-order" between configurations (see SMAC Sect. 5.2.2), but in the truly interesting models, such as spin glasses, no such trick is likely to exist. One must really supervise the 2^N configurations at $t = t_0$. This non-trivial task has been studied extensively (see Chanal and Krauth 2008).

2
Classical hard-sphere systems

2.1 Molecular dynamics

Before statistical mechanics, not so long ago, there only was classical mechanics. The junction between the two has fascinated generations of mathematicians and physicists, and nowhere can it be studied better than in the hard-sphere system: N particles in a box, moving about like billiard balls, with no other interaction than the hard-sphere exclusion (without friction or angular momentum). For more than a century, the hard-sphere model has been a prime example for statistical mechanics and a parade ground for rigorous mathematical physics. For more than fifty years, it has served as a test bed for computational physics, and it continues to play this role.

For concreteness, we first consider four disks (two-dimensional spheres) in a square box with walls (no periodic boundary conditions), as in Fig. 2.1: From an initial configuration, as the one at $t = 0$, particles fly apart on straight trajectories either until one of them hits a wall or until two disks undergo an elastic collision. The time for this next "event" can be computed exactly by going over all disks (taken by themselves, in the box, to compute the time for the next wall collisions) and all pairs of disks (isolated from the rest of the system, and from the walls, to determine the next pair collisions), and then treating the event closest in time (see SMAC Sect. 2.1).

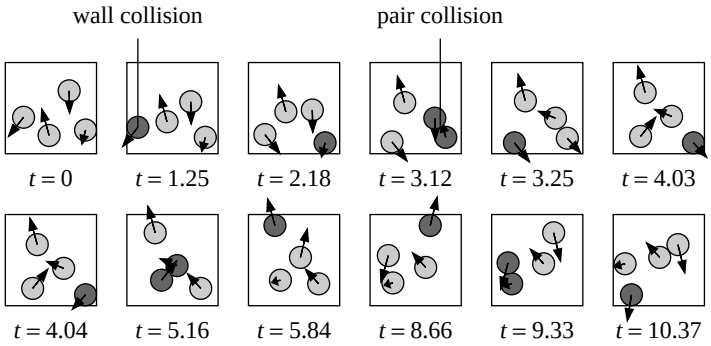

Fig. 2.1 Event-driven molecular dynamics simulation with four hard disks in a square box.

The event-chain algorithm can be implemented in a few dozen lines, just a few too many for a free afternoon in Les Houches (program listings are available on the SMAC web site). It implements the entire dynamics of the N-particle system without time discretization, because there is no differential equation to be solved. The only error

committed stems from the finite-precision arithmetic implemented on a computer.

This error manifests itself surprisingly quickly, even in our simulation of four disks in a square: typically, calculations done in 64bit precision (15 significant digits) get out of step with other calculations from identical initial conditions with 32bit calculations after a few dozen pair collisions. This is the manifestation of chaos in the hard-sphere system. The appearance of numerical uncertainties for given initial conditions can be delayed by using even higher precision arithmetic, but it is out of the question to control a calculation that has run for a few minutes on our laptop, and gone through a few billion collisions. Chaos in the hard sphere model has its origin in the negative curvature of the sphere surfaces, which magnifies tiny differences in the trajectory at each pair collision and causes serious roundoff errors in computations[1].

The mathematically rigorous analysis of chaos in the hard-disks system has met with resounding success: Sinai (1970) (for two disks) and Simanyi (2003, 2004) (for general hard-disk and hard-sphere systems) were able to mathematically prove the foundations of statistical physics for the system at hand, namely the equiprobability principe for hard spheres: This means that during an infinitely long molecular dynamics simulation, the probability density satisfies the following:

$$\left\{ \begin{array}{c} \text{probability of configuration with} \\ [\mathbf{x}_1, \mathbf{x}_1 + \mathrm{d}\mathbf{x}_1], \ldots, [\mathbf{x}_N, \mathbf{x}_N + \mathrm{d}\mathbf{x}_N] \end{array} \right\} \propto \pi(\mathbf{x}_1, \ldots, \mathbf{x}_N)\, \mathrm{d}\mathbf{x}_1, \ldots, \mathrm{d}\mathbf{x}_N,$$

where

$$\pi(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \begin{cases} 1 & \text{if configuration legal } (|\mathbf{x}_k - \mathbf{x}_l| > 2\sigma \text{ for } k \neq l) \\ 0 & \text{otherwise} \end{cases}. \tag{2.1}$$

An analogous property has been proven for the velocities:

$$\pi(\mathbf{v}_1, \ldots, \mathbf{v}_N) = \begin{cases} 1 & \text{if velocities legal } (\sum \mathbf{v}_k^2 = E_{\mathrm{kin}}/(2m)) \\ 0 & \text{otherwise} \end{cases}. \tag{2.2}$$

Velocities are legal if they add up to the correct value of the conserved kinetic energy, and their distribution is constant on the surface of the $2N$ dimensional hypersphere of radius $\sqrt{E_{\mathrm{kin}}/(2m)}$ (for disks).

The two above equations contain all of equilibrium statistical physics in a nutshell. The equal-probability principle of eqn (2.1) relates to the principle that two configurations of the same energy have the same statistical weight. The sampling problem for velocities, in eqn (2.2) can be solved with Gaussians, as discussed in SMAC Sect. 1.2.6. It reduces to the Maxwell distribution for large N (see SMAC Sect. 2.2.4) and implies the Boltzmann distribution (see SMAC Sect. 2.3.2).

2.2 Direct sampling, Markov chain sampling

We now move on from molecular dynamics simulations to the Monte Carlo sampling. To sample N disks with the constant probability distribution of eqn (2.1), we uniformly

[1] as it causes humiliating experiences at the billiard table

Algorithm 2.1 direct-disks.py

```
1  from random import uniform as ran
2  sigma=0.20
3  condition = False
4  while condition == False:
5    L = [(ran(sigma,1-sigma),ran(sigma,1-sigma))]
6    for k in range(1,4): # 4 particles considered
7      b = (ran(sigma,1-sigma),ran(sigma,1-sigma))
8      min_dist = min((b[0]-x[0])**2+(b[1]-x[1])**2 for x in L)
9      if min_dist < 4*sigma**2:
10       condition = False
11       break
12     else:
13       L.append(b)
14       condition = True
15 print L
```

throw a set of N particle positions $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ into the square. Each of these sets of N positions is generated with equal probability. We then sort out all those sets that are no legal hard-sphere configurations. The remaining ones (the gray configurations in Fig. 2.2) still have equal probabilities, exactly as called for in eqn (4.1). In the

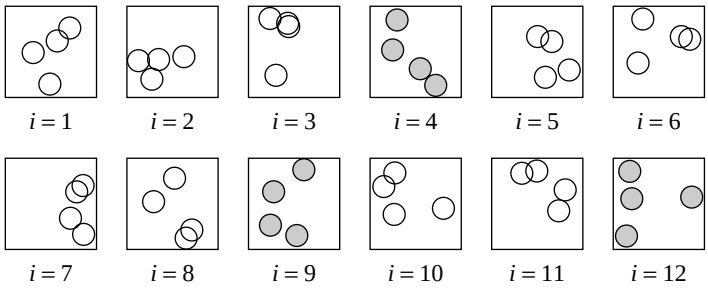

Fig. 2.2 Direct-sampling Monte Carlo algorithm for hard disks in a box without periodic boundary conditions (see Alg. `direct-disks.py`).

Python programming language, we can implement this algorithm in a few lines (see Alg. `direct-disks.py`): one places up to N particles at random positions (see line 7 of Alg. `direct-disks.py`). If two disks overlap, one breaks out of this construction and restarts with an empty configuration. The rejection rate $p_{\text{accept}}(N, \sigma)$ of this algorithm (the probability to generate legal (gray) configurations in Fig. 2.2):

$$p_{\text{accept}} = \frac{\text{number of valid configurations with radius } \sigma}{\text{number of valid configurations with radius } 0} = Z_\sigma / Z_0$$

is exponentially small both in the particle number N and in the density of particles, and this for physical reasons (see the discussion in SMAC Sect. 2.2.2)).

For the hard-sphere system, Markov-chain methods are much more widely applicable than the direct-sampling algorithm. In order to satisfy the detailed balance condition of eqn (4.2), we must impose that the probability to move from a legal con-

Algorithm 2.2 markov-disks.py

```
from random import uniform as ran, choice
L=[(0.25,0.25),(0.75,0.25),(0.25,0.75),(0.75,0.75)]
sigma,delta=0.20,0.15
for iter in range(1000000):
  a = choice(L)
  L.remove(a)
  b = (a[0] + ran(-delta,delta),a[1] + ran(-delta,delta))
  min_dist = min((b[0]-x[0])**2 + (b[1]-x[1])**2 for x in L)
  box_cond = min(b[0],b[1]) < sigma or max(b[0],b[1]) >1-sigma
  if box_cond or min_dist < 4*sigma**2:
   L.append(a)
  else:
   L.append(b)
print L
```

figuration a to another, b, must be the same as the probability to move from b back to a (see Fig. 2.3). This is realized most easily by picking a random disk and moving it inside a small square around its original position, as implemented in Alg. `markov-disks.py`.

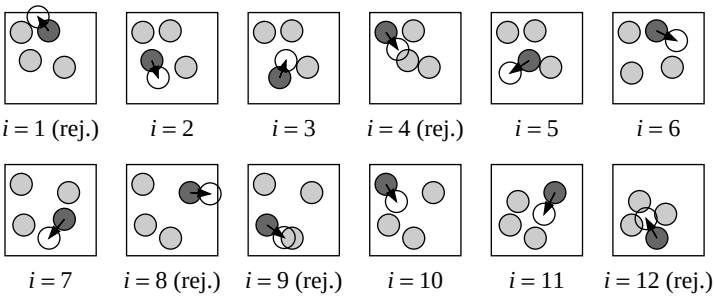

Fig. 2.3 Markov-chain Monte Carlo algorithm for hard disks in a box without periodic boundary conditions.

The Monte-Carlo dynamics of the Markov-chain algorithm, in Fig. 2.3, superficially resembles the molecular-dynamics in Fig. 2.1. Let us work out the essential differences between the two: In fact, the Monte Carlo dynamics is diffusive: It can be described in terms of diffusion constants and transfer matrices, and convergence towards the equilibrium distribution of eqn (2.1) is exponential. In contrast, the dynamics of the event-chain algorithm is hydrodynamic (it contains eddies, turbulence, etc, and their characteristic timescales). Although it converges to the same equilibrium state, as discussed before, this convergence is algebraic. This has dramatic consequences, especially in two dimensions (see SMAC Sect. 2.2.5), even though the long-time dynamics in a finite box is more or less equivalent in the two cases. This is the fascinating subject of long-time tails, discovered by Alder and Wainwright (1970), which for lack of time could not be covered in my lectures (see SMAC Sect. 2.2.5).

The local Markov-chain Monte Carlo algorithm runs into serious trouble at high density, where the Markov chain of configurations effectively gets stuck during long times (although it remains ergodic). The cleanest illustration of this fact is obtained

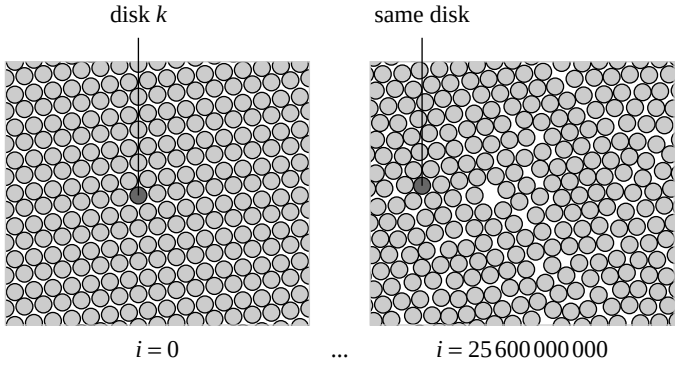

Fig. 2.4 Initial and final configurations for 256 disks at density $\eta = 0.72$ (much smaller than the close-packing density) after 25.6 billion iterations. The sample at iteration $i = 25.6 \times 10^9$ remembers the initial configuration.

by starting the run with an easily recognizable initial configuration, as the one shown in Fig. 2.4, which is slightly tilted with respect to the x-axis. We see in this example that $\tau_{\rm corr} \gg 2.56 \times 10^9$ iterations, even though it remains finite and can be measured in a Monte Carlo calculation.

Presently, no essentially faster algorithm than the local algorithm is known for uniform hard spheres, and many practical calculations (done with very large numbers of particles at high density) are clearly un-converged. Let us notice that the slowdown of the Monte Carlo calculation at high density has a well-defined physical origin: the slowdown of the single-particle diffusion at high density. However, this does not exclude the possibility of much faster algorithms, as we will discuss in the fourth lecture.

2.3 Cluster algorithms, birth-and-death processes

In the present section, we explore Monte Carlo algorithms that are not inspired by the physical process of single-particle diffusion underlying the local Markov-chain Monte Carlo algorithm. Instead of moving one particle after the other, cluster algorithms construct coordinated moves of several particles at a time, from one configuration, a, to a very different configuration, b in one deterministic step. The pivot cluster algorithm (Dress and Krauth 1995, Krauth and Moessner 2003) is the simplest representative of a whole class of algorithms.

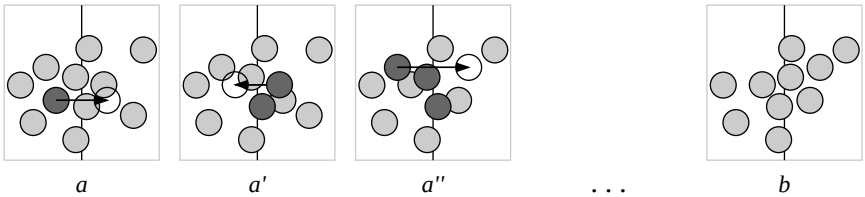

Fig. 2.5 A cluster move about a pivot axis involving five disks. The pivot-axis position and its orientation (horizontal, vertical, diagonal) are chosen randomly. Periodic boundary conditions allow us to scroll the pivot into the center of the box.

Algorithm 2.3 pocket-disks.py in a periodic box of size 1×1

```
from random import uniform as ran, choice
import box_it, dist # see SMAC web site for periodic distance
Others=[(0.25,0.25),(0.25,0.75),(0.75,0.25),(0.75,0.75)]
sigma_sq=0.15**2
for iter in range(10000):
    a = choice(Others)
    Others.remove(a)
    Pocket = [a]
    Pivot=(ran(0,1),ran(0,1))
    while Pocket != []:
        a = Pocket.pop()
        a = T(a,Pivot)
        for b in Others[:]: # "Others[:]" is a copy of "Others"
        if dist(a,b) < 4*sigma_sq:
            Others.remove(b)
            Pocket.append(b)
        Others.append(a)
print Others
```

In this algorithm, one uses a "pocket", a stack of disks that eventually have to move. Initially, the pocket contains a random disk. As long as there are disks in the pocket, one takes one of them out of it and moves it. It gets permanently placed, and all particles it overlaps with are added to the pocket (see Fig. 2.5, the "pocket particles" are colored in dark). In order to satisfy detailed balance, the move $a \to b = T(a)$ must have a Z_2 symmetry $a = T^2(a)$ (as a reflection around a point, an axis, or a hyperplane), such that moving it twice brings each particle back to its original position (see SMAC Sect. 2.5.2). In addition, the transformation T must map the simulation box onto itself. For example, we can use a reflection around a diagonal in a square or cube box, but not in a rectangle or cuboid. Periodic boundary conditions are an essential ingredient in this algorithm. In Python, the pocket algorithm can be implemented in a dozen lines of code (see Alg. `pocket-disks.py`).

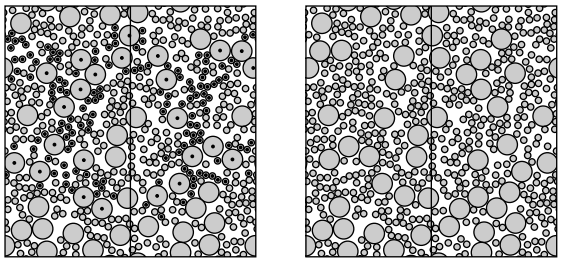

Fig. 2.6 Single iteration (from a to b) of the cluster algorithm for a binary mixture of hard spheres. The symmetry operation is a flip around the y axis shown. Transformed disks are marked with a dot (see Buhot and Krauth, 1999).

The pivot-cluster algorithm fails at high density, say, at the condition of Fig. 2.4, where the transformation T simply transforms all particles in the system. In that case

entire system without changing the relative particle positions. However, the algorithm is extremely efficient for simulations of monomer–dimer models or in binary mixtures, among others. An example of this is given in Fig. 2.6.

At the end of this lecture, let us formulate hard-sphere systems with a grand-canonical partition function and fugacity λ:

$$Z = \sum_{N=0}^{\infty} \lambda^N \pi(x_1, \ldots, x_N).$$

and discuss the related Markov-chain Monte Carlo algorithms in terms of birth-and-death processes (no existential connotation intended): Between two connected configurations, the configuration can only change through the appearance ("birth") or disappearance ("death") of a disk (see Fig. 2.7) It follows from the detailed balance

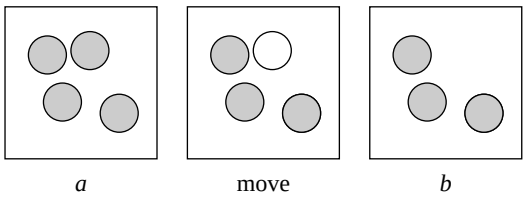

Fig. 2.7 Birth-and-death process for grand-canonical hard disks.

condition eqn (4.2) that the probability to $\pi(a) = \lambda\pi(b)$, so that $p(b \to a) = \lambda p(a \to b)$. This means that to sample eqn (2.3), we simply have to install one "death" probability (per time interval dt) for any particle, as well as a birth probability for creating a new particle anywhere in the system (this particle is rejected if it generates an overlap).

As in the so-called "faster-than-the-clock" algorithms (see SMAC Sect. 7.1) one does not discretize time $\tau \to \Delta_\tau$ but rather samples lifetimes and birth times from their exponential distributions. In Fig. 2.8, we show a time-space diagram of all the events that can happen in one extended simulation (as in the first lecture), in the coupling-from-the-past framework, as used by Kendall and Moller (2000). We do not know the configuration of disks but, on a closer look, we can deduce it, starting from the time t_0 indicated.

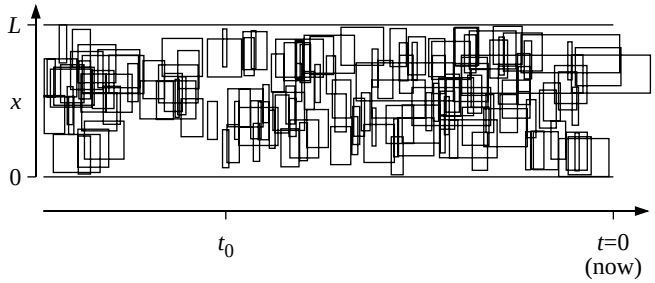

Fig. 2.8 One-dimensional time-space diagram of birth-and-death processes for hard disks in one dimension.

3
Quantum Monte Carlo simulations

3.1 Density matrices, naive quantum simulations

After the connection between classical mechanics and (classical) statistical mechanics, we now investigate the junction between statistical mechanics and quantum physics. We first consider a single particle of mass m in a harmonic potential $V(x) = \frac{1}{2}\omega x^2$, for which wave functions and energy eigenvalues are all known (see Fig. 3.1, we use units $\hbar = m = \omega = 1$).

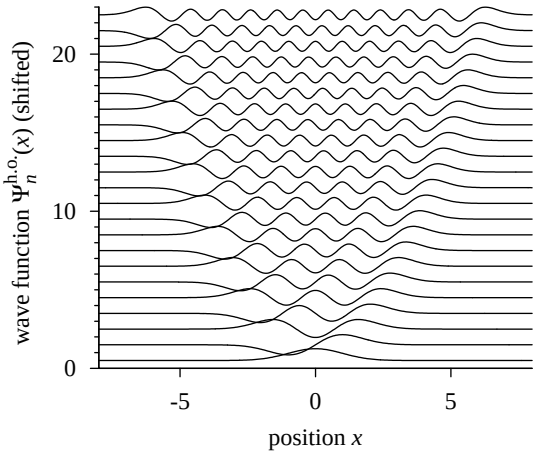

Fig. 3.1 Harmonic-oscillator wave functions ψ_n, shifted by the energy eigenvalue E_n.

As before, we are interested in the probability $\pi(x)$ for a particle to be at position x (compare with eqn (2.1)). This probability can be assembled from the statistical weight of level k, $\pi(E_k) \propto \exp(-\beta E_k)$ and from the quantum mechanical probability $\psi_k(x)\psi_k^*(x)$ to be at position x while in level k,

$$\pi(x) = \frac{1}{Z(\beta)} \underbrace{\sum_k \exp(-\beta E_k)\, \Psi_k(x)\Psi_k^*(x)}_{\rho(x,x,\beta)} . \tag{3.1}$$

This probability involves a diagonal element of the density matrix given by $\rho(x, x', \beta) = \sum_k \exp(-\beta E_k)\, \Psi_k(x)\Psi_k^*(x)$, whose trace is the partition function:

$$Z(\beta) = \int \mathrm{d}x\ \rho(x,x,\beta) = \sum_k \exp\left(-\beta E_k\right)$$

(see SMAC Sect 3.1.1). For the free particle and the harmonic oscillator, we can indeed compute the density matrix exactly (see the later eqn (3.5)) but in general, eigenstates or energies are out of reach for more complicated Hamiltonians. The path integral approach obtains the density matrix without knowing the spectrum of the Hamiltonian, by using the convolution property

$$\rho^{\mathrm{any}}(x,x',\beta) = \int \mathrm{d}x''\ \rho^{\mathrm{any}}(x,x'',\beta')\,\rho^{\mathrm{any}}(x'',x',\beta-\beta') \quad \text{(convolution)}, \tag{3.2}$$

which yields the density matrix at a given temperature through a product of density matrices at higher temperatures. In the high-temperature limit, the density matrix for the Hamiltonian $H = H^{\mathrm{free}} + V$ is given by the Trotter formula

$$\rho^{\mathrm{any}}(x,x',\beta) \xrightarrow[\beta\to 0]{} \mathrm{e}^{-\frac{1}{2}\beta V(x)}\rho^{\mathrm{free}}(x,x',\beta)\ \mathrm{e}^{-\frac{1}{2}\beta V(x')} \quad \text{(Trotter formula)}. \tag{3.3}$$

(compare with SMAC Sect. 3.1.2). For concreteness, we continue our discussion with the harmonic potential $V = \frac{1}{2}x^2$, although the discussed methods are completely general. Using eqn (3.2) repeatedly, one arrives at the path-integral representation of the partition function in terms of high-temperature density matrices:

$$Z(\beta) = \int \mathrm{d}x\ \rho(x,x,\beta) = \int \mathrm{d}x_0\ \ldots \int \mathrm{d}x_{N-1} \underbrace{\rho(x_0,x_1,\Delta_\tau)\ldots\rho(x_{N-1},x_0,\Delta_\tau)}_{\pi(x_0,\ldots,x_{N-1})} \tag{3.4}$$

where $\Delta_\tau = \beta/N$. In the remainder of this lecture we will focus on this multiple integral but we are again more interested in sampling (that is, in generating "paths" $\{x_0,\ldots,x_{N-1}\}$ with probability $\pi(x_0,\ldots,x_{N-1})$) than in actually computing it. A naive Monte Carlo sampling algorithm is set up in a few lines of code (see the Alg. `naive-harmonic-path.py`, on the SMAC web site). In the integrand of eqn (3.4), one chooses a random point $k \in [0, N-1]$ and a uniform random displacement $\delta_x \in [-\delta,\delta]$ (compare with Fig. 3.2). The acceptance probability of the move depends on the weights $\pi(x_0,\ldots,x_{N-1})$, thus both on the free density matrix part, and on the interaction potential.

The naive algorithm is extremely slow because it moves only a single "bead" k out of N, and not very far from its neighbors $k-1$ and $k+1$. At the same time, displacements of several beads at the same time would rarely be accepted. In order to go faster through configuration space, our proposed moves must learn about quantum mechanics. This is what we will teach them in the following section.

3.2 Direct sampling of a quantum particle: Lévy construction

The problem with naive path-sampling is that the algorithm lacks insight: the probability distribution of the proposed moves contains no information about quantum

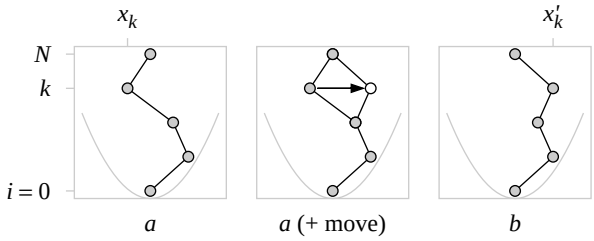

Fig. 3.2 Move of the path $\{x_0, \ldots, x_{N-1}\}$ *via* element $k \in [0, N-1]$, sampling the partition function in eqn (3.4). For $k = 0$, both 0 and N are displaced in the same direction. The external potential is indicated.

mechanics. However, this problem can be solved completely for a free quantum particle and also for a particle in a harmonic potential $V(x) = \frac{1}{2}x^2$, because in both cases the exact density matrix for a particle of mass $m = 1$, with $\hbar = 1$, is a Gaussian:

$$\rho(x, x', \beta) = \begin{cases} \frac{1}{\sqrt{2\pi\beta}} \exp\left[-\frac{(x-x')^2}{2\beta}\right] & \text{(free particle)} \\ \frac{1}{\sqrt{2\pi \sinh\beta}} \exp\left[-\frac{(x+x')^2}{4}\tanh\frac{\beta}{2} - \frac{(x-x')^2}{4}\coth\frac{\beta}{2}\right] & \text{(osc.)} \end{cases} \tag{3.5}$$

The distribution of an intermediate point, for example in the left panel of Fig. 3.3, is given by

$$\pi(x_1|x_0, x_N) = \underbrace{\rho\left(x_0, x_1, \frac{\beta}{N}\right)}_{\text{Gaussian, see eqn (3.5)}} \underbrace{\rho\left(x_1, x_N, \frac{(N-1)\beta}{N}\right)}_{\text{Gaussian, see eqn (3.5)}}.$$

As a product of two Gaussians, this is again a Gaussian, and it can be sampled directly. After sampling x_1, one can go on to x_2, etc., until the the whole path is constructed, and without rejections (Lévy 1940).

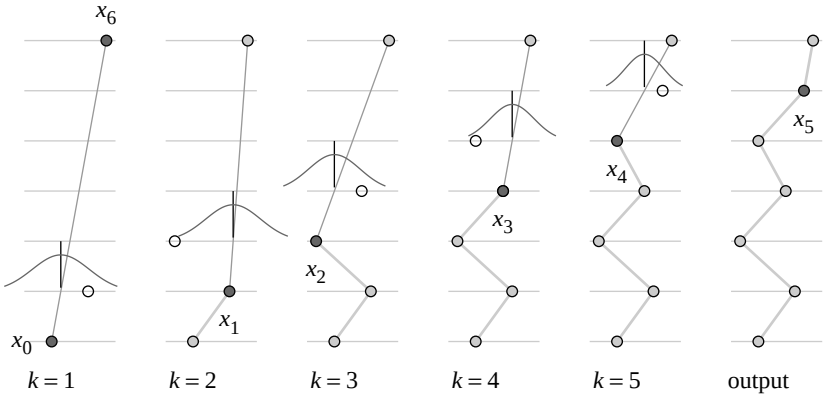

Fig. 3.3 Lévy construction of a path contributing to the free-particle density matrix $\rho(x_0, x_6, \beta)$. The intermediate distributions, all Gaussians, can be sampled directly.

Algorithm 3.1 levy-free-path.py

```
from math import pi, exp, sqrt
from random import gauss
N = 10000
beta = 1.
Del_tau = beta/N
x = [0 for k in range(N+1)]
y = range(N+1)
for k in range(1,N):
   Del_p = (N-k)*Del_tau
   x_mean = (Del_p*x[k-1] + Del_tau*x[N])/(Del_tau + Del_p)
   sigma = sqrt(1./(1./Del_tau + 1./Del_p))
   x[k] = gauss(x_mean,sigma)
print x
```

The Lévy construction is exact for a free particle in a harmonic potential and of course also for free particles (as shown in the Python code). Direct-sampling algorithms can be tailored to specific situations, such as periodic boundary conditions or hard walls (see SMAC Sections 3.3.2 and 3.3.3). Moreover, even for generic interactions, the Lévy construction allows to pull out, treat exactly, and sample without rejections the free-particle Hamiltonian. In the generic Hamiltonian $H = H_0 + V$, the Metropolis rejection then only takes care of the interaction term V. Much larger moves than before become possible and the phase space is run through more efficiently than in the naive algorithm, where the Metropolis rejection concerned the entire H. This makes possible nontrivial simulations with a very large number of interacting particles (see Krauth 1996, Holzmann and Krauth 2008). We will illustrate this point in the following section, showing that N-body simulations of ideal (non-interacting) bosons (of arbitrary size) can be done by direct sampling without any rejection.

We note that the density matrices in eqn (3.5), for a single particle in a d-dimensional harmonic oscillator yield the partition functions

$$z^{\text{h.o.}}(\beta) = \int \mathrm{d}x \rho(x, x, \beta) = [1 - \exp(-\beta)]^{-d} \quad (\text{with } E_0 = 0) \tag{3.6}$$

where we use the lowercase symbol $z(\beta)$ in order to differentiate the one-particle partition function from its N-body counterpart $Z(\beta)$ which we will need in the next section.

3.3 Ideal bosons: Landsberg recursion and direct sampling

The density matrix for N distinguishable particles $\rho^{\text{dist}}(\{x_1, \ldots, x_N\}, \{x'_1, \ldots, x'_N\}, \beta)$ is assembled from normalized N-particle wavefunctions and their associated energies, as in eqn (3.1). The density matrix for indistinguishable particles is then obtained by symmetrizing the density matrix for N distinguishable particles. This can be done either by using symmetrized (and normalized) wavefunctions to construct the density matrix or, equivalently, by averaging the distinguishable-particle density matrix over all $N!$ permutations (Feynman 1972):

$$Z = \frac{1}{N!} \sum_P \int \mathrm{d}^N x \rho^{\mathrm{dist}}\big(\{x_1, \ldots, x_N\}, \{x_{P(1)}, \ldots, x_{P(N)}\}, \beta\big) \,. \tag{3.7}$$

This equation is illustrated in Fig. 3.4 for four particles. In each diagram, we arrange the points x_1, x_2, x_3, x_4 from left to right and indicate the permutation by lines. The final permutation (to the lower right of Fig. 3.4), corresponds for example to the permutation $P(1,2,3,4) = (4,3,2,1)$. It consists of two cycles of length 2, because $P^2(1) = P(4) = 1$ and $P^2(2) = P(3) = 2$.

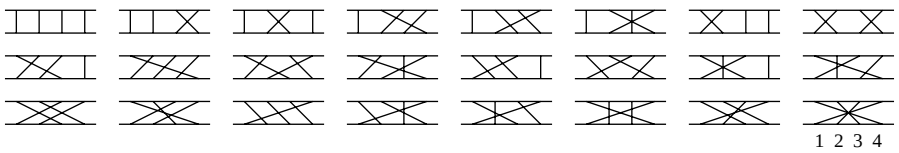

Fig. 3.4 The 24 permutations contributing to the partition function for 4 ideal bosons.

To illustrate eqn (3.7), let us compute the contribution to the partition function stemming from this permutation for free particles:

$$Z_{4321} = \int \mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}x_3 \mathrm{d}x_4 \rho^{\mathrm{free}}(\{x_1, x_2, x_3, x_4\}, \{x_4, x_3, x_2, x_1\}, \beta) =$$
$$\underbrace{\left[\int \mathrm{d}x_1 \mathrm{d}x_4 \rho^{\mathrm{free}}(x_1, x_4, \beta)\, \rho^{\mathrm{free}}(x_4, x_1, \beta)\right]}_{\int \mathrm{d}x_1 \rho^{\mathrm{free}}(x_1, x_1, 2\beta) = z(2\beta) \quad \text{see eqn (3.2)}} \left[\int \mathrm{d}x_2 \mathrm{d}x_3 \rho^{\mathrm{free}}(x_2, x_3, \beta)\, \rho^{\mathrm{free}}(x_3, x_2, \beta)\right] .$$

This partial partition function of a four-particle system writes as the product of one-particle partition functions. The number of terms corresponds to the number of cycles and the length of cycles determines the effective inverse temperatures in the systems (Feynman 1972).

The partition function for 4 bosons is given by a sum over 24 terms, 5 bosons involve 120 terms, and the number of terms in the partition function for a million bosons has more than 5 million digits. Nevertheless, this sum over permutations can be computed exactly through an ingenious recursive procedure of N steps due to Landsberg (1961) (see Fig. 3.5).

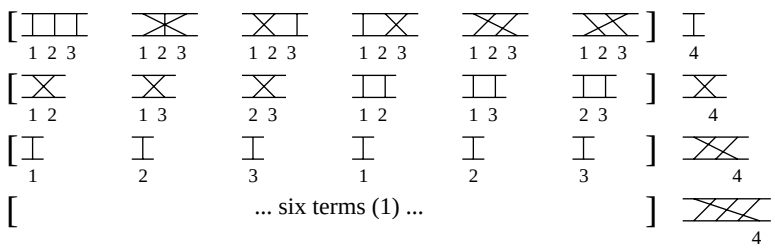

Fig. 3.5 The 24 permutations of Fig. 3.4, with the last-element cycle pulled out of the diagram.

Algorithm 3.2 canonic-recursion.py

```
import math
def z(beta,k):
    sum = 1/(1- math.exp(-k*beta))**3
return sum
N=1000
beta=1./5
Z=[1.]
for M in range(1,N+1):
    Z.append(sum(Z[k] * z(beta,M-k) for k in range(M))/M)
pi_list=[z(beta,k)*Z[N-k]/Z[N]/N for k  in range(1,N+1)]
```

Figure 3.5 contains the same permutations as Fig. 3.4, but they have been rearranged and the last-element cycle (the one containing the particle 4) has been pulled out. These pulled-out terms outside braces make up the partition function of a single particle at temperature β (on the first row), at inverse temperature 2β (on the second row), etc., as already computed in eqn (3.6). The diagrams within braces in Fig. 3.5 contain the $3! = 6$ three-boson partition functions making up Z_3 (on the first line of the figure). The second row contains three times the diagrams making up Z_2, etc. All these terms yield together the terms in eqn (3.8):

$$Z_N = \frac{1}{N}\left(z_1 Z_{N-1} + \cdots + z_k Z_{N-k} + \cdots + z_N Z_0\right) \quad \text{(ideal Bose gas)} \tag{3.8}$$

(with $Z_0 = 1$, see SMAC Sect. 4.2.3 for a proper derivation). Z_0, as well as the single-particle partition functions are known from eqn (3.6), so that we can first determine Z_1, then Z_2, and so on (see the Alg. `harmonic-recursion.py`, the SMAC web site contains a version with graphics output).

The term in the Landsberg relation of eqn (3.8) $\propto z_k Z_{N-k}$ can be interpreted as a cycle weight, the statistical weight of all permutations in which the particle N is in a cycle of length k

$$\pi_k = \frac{1}{N Z_N} z_k Z_{N-k} \quad \text{(cycle weight)}.$$

From the Landsberg recursion, we can explicitly compute cycle weights for arbitrary large ideal-boson systems at any temperature (see Fig. 3.6).

In Fig. 3.6, we notice that the cycle-weight distribution π_k is flat for most k before it drops to zero around $k \sim 780$. Curiously, the derivative of this function yields the distribution of the condensate fraction (Holzmann and Krauth 1999, Chevallier and Krauth 2007), so that we see that at the temperature chosen, there are about 780 particles in the groundstate ($N_0/N \simeq 0.78$). At higher temperatures, the distribution of cycle weights is narrower. This means that there are no long cycles.

We now turn the eqn (3.8) around, as we first did for the Gaussian integral: rather than computing $Z(\beta)$ from the cycle weights, we sample the cycle distribution from its weights that is, pick one of the π_k with

$$\pi_1, \ldots, \pi_k, \ldots, \pi_N.$$

(we pick a cycle length 1 with probability π_1, 2 with probability π_2, and so on (it is best to use the tower-sampling algorithm we mentioned in the first lecture). Suppose

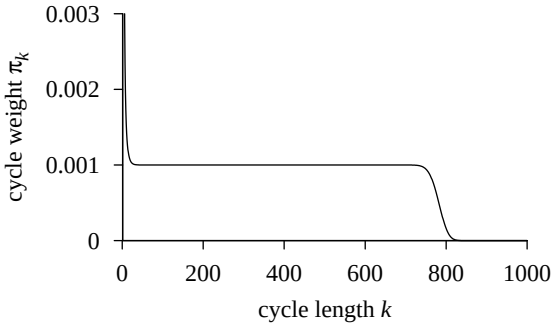

Fig. 3.6 Cycle-weight distribution for 1000 ideal bosons in a three-dimensional harmonic trap, obtained from Alg. `canonic-recursion.py`, exactly as written

we sampled a cycle length k. We then know that our Bose gas contains a cycle of length k, and we can sample the three-dimensional positions of the k particles on this cycle from the three-dimensional version of Alg. `harmonic-levy.py` for k particles instead of N, and at inverse temperature $k\beta$. Thereafter, we sample the next cycle length from the Landsberg recursion relation with $N-k$ particles instead of N, and so on, until all particles are used up (see the Appendix for a complete program in 44 lines). Output of this program is shown in Fig. 3.7, projected onto two dimensions. As in a real experiment, the three-dimensional harmonic potential confines the particles but, as we pass the Bose–Einstein transition temperature (roughly at the temperature of the left panel of Fig. 3.7), they start to move to the center and to produce the landmark peak in the spatial density. At the temperature of the right-side panel, roughly 80% of particles are condensed into the ground state.

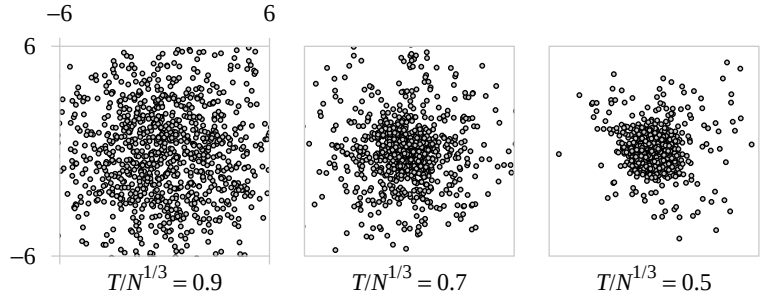

Fig. 3.7 Two-dimensional snapshots of 1000 ideal bosons in a three-dimensional harmonic trap (obtained with the direct-sampling algorithm).

The power of the path-integral approach resides in the facility with which interactions can be included. This goes beyond what can be treated in an introductory lecture (see SMAC Sect. 3.4.2 for an in-depth discussion). For the time being we should take pride in our rudimentary sampling algorithm for ideal bosons, a true quantum Monte Carlo program in a nutshell.

4

Spin systems: samples and exact solutions

4.1 Ising Markov-chains: local moves, cluster moves

In this final lecture, we study models of discrete spins $\{\sigma_1, \ldots, \sigma_N\}$ with $\sigma_k = \pm 1$ on a lattice with N sites, with energy

$$E = -\sum_{\langle k,l \rangle} J_{kl}\sigma_k\sigma_l. \tag{4.1}$$

Each pair of neighboring sites k and l is counted only once. In eqn (4.1), we may choose all the J_{kl} equal to $+1$. We then have the ferromagnetic Ising model. If we choose random values $J_{kl} = J_{lk} = \pm 1$, one speaks of the Edwards–Anderson spin glass model. Together with the hard-sphere systems, these models belong to the hall of fame of statistical mechanics, and have been the crystallization points for many developments in computational physics. Our goal in this lecture will be two-fold. We shall illustrate several algorithms for simulating Ising models and Ising spin glasses in a concrete setting. We shall also explore the relationship between the Monte Carlo sampling approach and analytic solutions, in this case the analytic solution for the two-dimensional Ising model initiated by Onsager (1942), Kac and Ward (1952), and Kaufmann (1949).

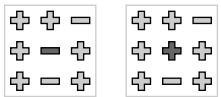

Fig. 4.1 A spin flip in the Ising model. Configuration a has central field $h = 2$ (three "+" neighbors and one "−" neighbor), and configuration b has $h = -2$. The statistical weights satisfy $\pi_b/\pi_a = \exp(-4\beta)$.

The simplest Monte Carlo algorithm for sampling the partition function

$$Z = \sum_{\text{confs } \boldsymbol{\sigma}} \exp\left[-\beta E(\boldsymbol{\sigma})\right]$$

picks a site at random, for example the central site on the square lattice of configuration a in Fig. 4.1. Flipping this spin would produce the configuration b. To satisfy detailed balance, eqn (4.2), we must accept the move $a \to b$ with probability $\min(1, \pi_b/\pi_a)$, as

Algorithm 4.1 markov-ising.py

```
from random import uniform as ran, randint,choice
from math import exp
import square_neighbors # defines the neighbors of a site
L = 32
N = L*L
S = [choice([-1,1]) for k in range(N)]
beta = 0.42
nbr = square_neighbors(L)
# nbr[k] = (right,up,left,down)
for i_sweep in range(100):
    for iter in range(N):
        k = randint(0,N-1)
        h = sum(S[nbr[k][j]] for j in range(4))
        Delta_E = 2.*h*S[k]
        Upsilon = exp(-beta*Delta_E)
        if ran(0.,1.) < Upsilon: S[k] = -S[k]
print S
```

implemented in the program Alg. `markov-ising.py` (see SMAC Sect. 5.2.1). This algorithm is very slow, especially near the phase transition between the low-temperature ferromagnetic phase and the high-temperature paramagnet. This slowdown is due to the fact that, close to the transitions, configurations with very different values of the total magnetization $M = \sum_k \sigma_k$ contribute with considerable weight to the partition function (in two dimension, the distribution ranges practically from $-N$ to N). One step of the local algorithm changes the total magnetization at most by a tiny amount, ± 2, and it thus takes approximately N^2 steps (as in a random walk) to go from one configuration to an independent one. This, in a nutshell, is the phenomenon of critical slowing down.

One can overcome critical slowing down by flipping a whole cluster of spins simultaneously, using moves that know about statistical mechanics (Wolff 1989). It is best to start from a random site, and then to repeatedly add to the cluster, with probability p, the neighbors of sites already present if the spins all have the same sign. At the end of the construction, the whole cluster is flipped. We now compute the value of p for which this algorithm has no rejections (see SMAC Sect. 5.2.3).

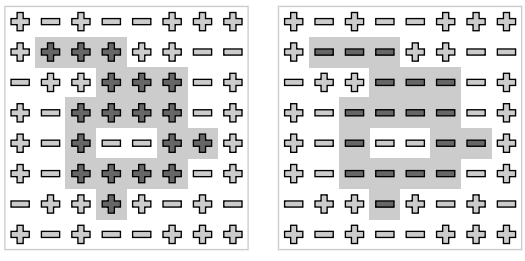

Fig. 4.2 A large cluster of like spins in the Ising model. The construction stops with the gray cluster in a with probability $(1-p)^{14}$, corresponding to the 14 "++" links across the boundary. The corresponding probability for b is $(1-p)^{18}$.

The cluster construction stops with the configuration a of Fig. 4.2 with probability $p(a \to b) = \text{const}(1-p)^{14}$, one factor of $(1-p)$ for every link "++" across the cluster boundary. Likewise, the cluster construction in b stops as shown in b if the 18 neighbors "−−" were considered without success (this happens with probability $p(b \to a) = \text{const}(1-p)^{18}$). The detailed balance condition relates the construction probabilities to the Boltzmann weights of the configurations

$$\pi_a = \text{const}' \exp\left[-\beta(-14+18)\right]$$
$$\pi_b = \text{const}' \exp\left[-\beta(-18+14)\right],$$

where the const′ describes all the contributions to the energy not coming from the boundary. We may enter stopping probabilities and Boltzmann weights into the detailed balance condition $\pi_a p(a \to b) = \pi_b \pi(b \to a)$ and find

$$\exp(14\beta)\exp(-18\beta)(1-p)^{14} = \exp(-14\beta)\exp(18\beta)(1-p)^{18}. \tag{4.2}$$

This is true for $p = 1 - \exp(-2\beta)$, and is independent of our example, with its 14 "++" and 18 neighbors "−−" links (see SMAC Sect. 4.2.3).

The cluster algorithm is implemented in a few lines (see Alg. `cluster-ising.py`) using the pocket approach of Section 2.3: Let the "pocket" comprise those sites of the cluster whose neighbors have not already been scrutinized. The algorithm starts by putting a random site both into the cluster and the pocket. One then takes a site out of the pocket, and adds neighboring sites (to the cluster and to the pocket) with probability p if their spins are the same, and if they are not already in the cluster. The construction ends when the pocket is empty. We then flip the cluster.

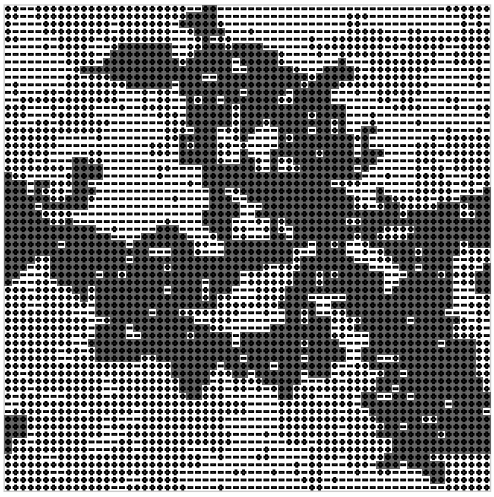

Fig. 4.3 A large cluster with 1548 spins in a 64×64 Ising model with periodic boundary conditions. All the spins in the cluster flip together from "+" to "−".

Algorithm 4.2 cluster-ising.py

```
from random import uniform as ran, randint,choice
from math import exp
import square_neighbors # defines the neighbors of a site
L = 32
N = L*L
S = [choice([-1,1]) for k in range(N)]
beta = 0.4407
p = 1-exp(-2*beta)
nbr = square_neighbors(L) # nbr[k]= (right,up,left,down)
for iter in range(100):
    k = randint(0,N-1)
    Pocket = [k]
    Cluster = [k]
    while Pocket != []:
        k = choice(Pocket)
        for l in nbr[k]:
            if S[l] == S[k] and l not in Cluster and ran(0,1) < p:
                Pocket.append(l)
                Cluster.append(l)
        Pocket.remove(k)
    for k in Cluster: S[k] = -S[k]
print S
```

Cluster methods play a crucial role in computational statistical physics because, unlike local algorithms and unlike experiments, they do not suffer from critical slowing down. These methods have spread from the Ising model to many other fields of statistical physics.

Today, the non-intuitive rules for the cluster construction are well understood, and the algorithms are very simple. In addition, the modern meta languages are so powerful that a rainy Les Houches afternoon provides ample time to implement the method, even for a complete non-expert in the field.

4.2 Perfect sampling: semi-order and patches

The local Metropolis algorithm picks a site at random and flips it with the probability $\min(1, \pi_b/\pi_a)$ (as in Section 4.1). An alternative local Monte Carlo scheme is the heatbath algorithm, where the spin is equilibrated in its local environment (see Fig. 4.4). This means that in the presence of a molecular field h at site k, the spin points up and down with probabilities π_h^+ and π_h^-, respectively, where

$$\begin{aligned} \pi_h^+ &= \frac{\mathrm{e}^{-\beta E^+}}{\mathrm{e}^{-\beta E^+} + \mathrm{e}^{-\beta E^-}} = \frac{1}{1+\mathrm{e}^{-2\beta h}}, \\ \pi_h^- &= \frac{\mathrm{e}^{-\beta E^-}}{\mathrm{e}^{-\beta E^+} + \mathrm{e}^{-\beta E^-}} = \frac{1}{1+\mathrm{e}^{+2\beta h}}. \end{aligned} \tag{4.3}$$

The heatbath algorithm (which is again much slower than the cluster algorithm, especially near the critical point) couples just like our trivial simulation in the five-site model of Section 1.2: At each step, we pick a random site k, and a random number

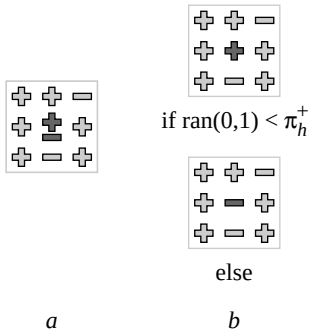

Fig. 4.4 Heat bath algorithm for the Ising model. The new position of the spin k (in configuration b) is independent of its original position (in a).

$\Upsilon = \texttt{ran}\,(0,1)$, and apply the Monte Carlo update of Fig. 4.4 with the same k and the same Υ to all the configurations of the Ising model or spin glass. After a time $\tau_{\rm coup}$, all input configurations yield identical output (Propp and Wilson 1996).

For the Ising model (but not for the spin glass) it is very easy to compute the coupling time, because the half-order among spin configurations is preserved by the heat-bath dynamics (see Fig. 4.5): we say that a configuration $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_N\}$ is smaller than another configuration $\boldsymbol{\sigma}' = \{\sigma'_1, \ldots, \sigma'_N\}$ if for all k we have $\sigma_k \leq \sigma'_k$. [1] For the Ising model, the heat-bath algorithm preserves the half-order between configurations upon update because a configuration which is smaller than another one has a smaller field on all sites, thus a smaller value of π_h^+. We just have to start the simulation from the all plus-polarized and the all minus-polarized configurations and wait until these two extremal configurations couple. This determines the coupling time for all 2^N configurations, exactly as in the earlier trivial example of Fig. 1.4.

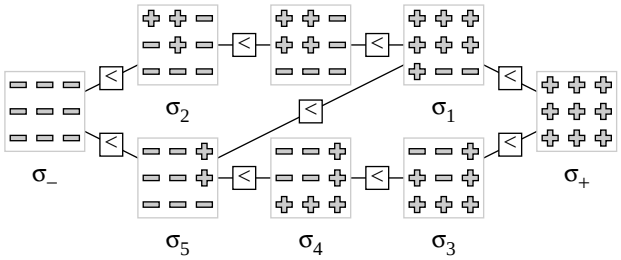

Fig. 4.5 Half-order in spin models: the configuration $\boldsymbol{\sigma}_-$ is smaller and $\boldsymbol{\sigma}_+$ is larger than all other configurations, which are not all related among each other.

The Ising spin glass does not allow such simple tricks to be played, and we must explicitly check the coupling for all 2^N initial configurations. This enormous surveying task can be simplified considerably by breaking up the configurations on the entire lattice into smaller "patches" (see Fig. 4.6). For a given patch size M ($M = 16$ in Fig. 4.6), the total number of configurations on N patches is bounded by $N2^M \ll 2^N$.

[1] This is a half-order because not all configurations can be compared to each other.

We can now follow the heatbath simulation on all individual patches and at the end assemble configurations on the whole lattice in the same way as we assemble an entire puzzle picture from the individual pieces. The procedure can be made practical, and programmed very easily in modern meta languages such as Python.

From a more fundamental point of view, the coupling concept in spin glass models is of interest because the physical understanding of these systems has seriously suffered from longstanding doubts about the quality of Monte Carlo calculations thus, at the most basic level, about the correct calculations of the correlation time.

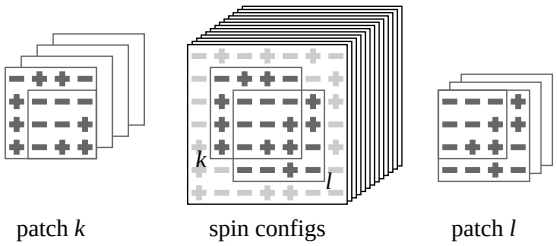

Fig. 4.6 The very large configuration space of an Ising spin glass on N sites viewed from the vantage point of N patches of smaller size M.

4.3 Direct sampling, and the Onsager solution

The two-dimensional Ising model is exactly solvable, as shown by Onsager (1942): we can compute its critical temperature, and many critical exponents. Let us be more precise: The partition function of the two-dimensional Ising model or the spin glass[2] on a planar lattice with N sites can be expressed as the square root of the determinant of a $4N \times 4N$ matrix (see SMAC Sect. 5.1.4 for a practical algorithm). Periodic boundary conditions can also be handled, it gives four $4N \times 4N$ matrices. This was used very successfully by Saul and Kardar (1992). For the Ising model on a finite square lattice, these matrices can be diagonalized analytically. This is the famous analytic expression for $Z(\beta)$ of Kaufman (1949) (see SMAC exerc. 5.9, p. 265).

The solution of the Ising model amounts to summing its high-temperature series, and this solves an enumeration problem, as evidenced in the combinatorial solution by Kac and Ward (1949). Indeed, the density of states of the energy $\mathcal{N}(E)$ can be extracted from the analytic solution (see Beale (1996)), this means that we can obtain the (integer) number of states for any energy E for two-dimensional Ising spin glasses of very large sizes. The exact solution of the Ising model thus performs an enumeration, but it counts configurations, it cannot list them. The final point we elaborate on in these lectures is that, unable to list configurations, we can still sample them.

The subtle difference between counting and listing of sets implies that while we access without any trouble the density of states $\mathcal{N}(E)$ we cannot obtain the distribution (the histogram) of magnetizations and know very little about the joint distribution of

[2] we speak here of one single "sample" of the spin glass, that is, a given choice of the couplings $\{J_{kl}\}$. To compute the average over all couplings is another matter.

energies and magnetizations, $\mathcal{N}(M, E)$. Doing so would in fact allow us to solve the Ising model in a magnetic field, which is not possible.

Let us now expose a sampling algorithm (Chanal and Krauth 2009), which uses the analytic solution of the Ising model to compute $\mathcal{N}(M, E)$ (with only statistical, and no systematic errors) even though that is impossible to do (exactly). This algorithm constructs the sample one site after another. Let us suppose that the gray spins in the left panel of Fig. 4.7 are already fixed, as shown. We can now set a fictitious coupling J^*_{ll} either to $-\infty$ or two $+\infty$ and recalculate the partition function $Z_\pm$ with them. The statistical weight of all configurations in the original partition function with spin "+" is then given by

$$\pi_+ = \frac{Z_+ \exp(\beta J_{kl})}{Z_+ \exp(\beta J_{kl}) + Z_- \exp(-\beta J_{kl})}. \tag{4.4}$$

and this two-valued distribution can be sampled with one random number. Equation (4.4) resembles the heatbath algorithm of eqn (4.3), but it is not part of a Markov chain: After obtaining the value of the spin on site k, we keep the fictitious coupling, and add more sites. Going over all sites, we can generate direct samples of the partition function of eqn (4.1), at any temperature, and with a fixed, temperature-independent effort, both for the two-dimensional Ising model and the Ising spin glass. This allows us to obtain arbitrary correlation functions, the generalized density of states $\mathcal{N}(M, E)$, etc. We can also use this solution to determine the behavior of the Ising model in a magnetic field, which we cannot obtain from the analytic solution, although we use it in the algorithm.

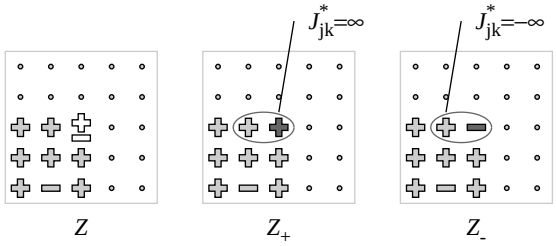

Fig. 4.7 One iteration in the direct-sampling algorithm for the two-dimensional Ising model. The probabilities $\pi(\sigma_k = +1)$ and $\pi(\sigma_k = -1)$ (with k the central spin) are obtained from the exact solution of the Ising model with fictitious couplings $J^*_{jk} = \pm\infty$.

The correlation-free direct-sampling algorithm is primarily of theoretical interest, as it is restricted to two-dimensional Ising model and the Ising spin glass, which are already well understood. For example, it is known that the two-dimensional spin glass does not have a finite transition temperature. Nevertheless, it is intriguing that one can obtain, from the analytical solution, exactly and with a performance guarantee, quantities that the analytical solution cannot give. The "sampling" (even the very well controlled, full-performance-guarantee direct sampling used here) does not meet the limitations of complete enumerations.

Acknowledgements

I would like to thank the organizers of this School for giving me the opportunity to lecture about some of my favorite subjects. Thanks to C. Laumann for introducing me to Python programming and for pointing out the similarities with SMAC pseudocode and to C. Chanal and M. Chevallier for a careful reading of the manuscript.

Appendix A

Here is an entire Quantum Monte Carlo program (in Python 2.5)) for ideal bosons in a harmonic trap (see the SMAC web site for a program with graphics output).

Algorithm A.1 direct-harmonic-bosons.py

```
from math import sqrt, sinh, tanh, exp
from random import uniform as ran, gauss
def z(beta,k):
    sum = 1/(1-exp(-k*beta))**3
    return sum
def canonic_recursion(beta,N):
    Z=[1.]
    for M in range(1,N+1):
        Z.append(sum(Z[k]*z(beta,M-k) for k in range(M))/M)
    return Z
def pi_list_make(Z,M):
    pi_list=[0]+[z(beta,k)*Z[M-k]/Z[M]/M for k in range(1,M+1)]
    pi_sum=[0]
    for k in range(1,M+1):
        pi_sum.append(pi_sum[k-1]+pi_list[k])
    return pi_sum
def tower_sample(data,Upsilon): #naive, cf. SMAC Sect. 1.2.3
    for k in range(len(data)):
        if Upsilon<data[k]: break
    return k
def levy_harmonic_path(Del_tau,N): #
    beta=N*Del_tau
    xN=gauss(0.,1./sqrt(2*tanh(beta/2.)))
    x=[xN]
    for k in range(1,N):
        Upsilon_1 = 1./tanh(Del_tau)+1./tanh((N-k)*Del_tau)
        Upsilon_2 = x[k-1]/sinh(Del_tau)+xN/sinh((N-k)*Del_tau)
        x_mean=Upsilon_2/Upsilon_1
        sigma=1./sqrt(Upsilon_1)
        x.append(gauss(x_mean,sigma))
    return x
N=512
beta=1./2.4
Z=canonic_recursion(beta,N)
M=N
x_config=[]
y_config=[]
while M > 0:
    pi_sum=pi_list_make(Z,M)
    Upsilon=ran(0,1)
    k=tower_sample(pi_sum,Upsilon)
    x_config+=levy_harmonic_path(beta,k)
    y_config+=levy_harmonic_path(beta,k)
    M -= k
```

References

Alder B. J., Wainwright T. E. (1970) Decay of the velocity autocorrelation function, *Physical Review A* **1**, 18–21

Beale P. D. (1996) Exact distribution of energies in the two-dimensional Ising model, *Physical Review Letters* **76**, 78–81

Buhot A., Krauth W. (1999) Phase separation in two-dimensional additive mixtures, *Physical Review E* **59**, 2939–2941

Chanal C., Krauth, W. (2008) Renormalization group approach to exact sampling, *Physical Review Letters* **100**, 060601

Chanal C., Krauth, W. (2009) manuscript in preparation

Chevallier M., Krauth W. (2007) Off-diagonal long-range order, cycle probabilities, and condensate fraction in the ideal Bose gas, *Physical Review E* **76**, 051109

Dress C., Krauth W. (1995) Cluster algorithm for hard spheres and related systems, *Journal of Physics A* **28**, L597–L601

Feynman R. P. (1972) *Statistical Mechanics: A Set of Lectures*, Benjamin/Cummings, Reading, Massachusetts

Holzmann M., Krauth W. (1999) Transition temperature of the homogeneous, weakly interacting Bose gas, *Physical Review Letters* **83**, 2687

Kaufman B. (1949) Crystal Statistics. II. Partition function evaluated by spinor analysis, *Physical Review* **76**, 1232–1243

Kendall W. S., Moller J. (2000) Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes, *Advances in applied probability* **32**, 844–865

Krauth W. (1996) Quantum Monte Carlo calculations for a large number of bosons in a harmonic trap, *Physical Review Letters* **77**, 3695–3699

Krauth W. (2006) *Statistical Mechanics: Algorithms and Computations*, Oxford University Press, Oxford

Krauth W., Moessner R. (2003) Pocket Monte Carlo algorithm for classical doped dimer models, *Physical Review B* **67**, 064503

Landsberg P. T. (1961) *Thermodynamics, with quantum statistical illustrations*, Interscience, New York

Lévy P. (1940) Sur certains processus stochastiques homogènes [in French], *Composition Mathematica* **7**, 283–339

Onsager L. (1944) Crystal Statistics. I. A two-dimensional model with an order-disorder transition, *Physical Review* **65**, 117–149

Propp J. G., Wilson D. B. (1996) Exact sampling with coupled Markov chains and applications to statistical mechanics, *Random Structures & Algorithms* **9**, 223–252

Saul L., Kardar M. (1993) Exact integer algorithm for the two-dimensional $\pm J$ Ising spin glass, *Physical Review E* **48**, R3221–R3224

Simanyi N. (2003) Proof of the Boltzmann–Sinai ergodic hypothesis for typical hard disk systems, *Inventiones Mathematicae* **154**, 123–178

Simanyi N. (2004) Proof of the ergodic hypothesis for typical hard ball systems, *Annales de l'Institut Henri Poincaré* **5**, 203–233

Sinai Y. G. (1970) Dynamical systems with elastic reflections, *Russian Mathematical Surveys* **25**, 137–189

Wolff U. (1989) Collective Monte-Carlo updating for spin systems, *Physical Review Letters* **62**, 361–364