

### 3. Importance sampling and update algorithms

#### 3.1. Canonical ensemble

- One of the most common use for Monte Carlo method is to study *thermodynamics* of some statistical model defined on a lattice. For example, we may have a (real-valued, say) field  $\phi_x$ , which is defined at each lattice coordinate  $x$ , with the Hamiltonian energy functional  $H(\phi)$ . We shall discuss here regular (hyper) cubic lattices, where

$$x_i = a n_i, \quad n_i = 0 \dots N_i,$$

where  $a$  is the lattice spacing (in physical units).

- The most common statistical ensemble we meet is the *canonical ensemble*, which is defined by the partition function

$$Z = \int [d\phi] \exp \left[ -\frac{H(\phi)}{k_B T} \right].$$

Here  $[d\phi] \equiv \prod_x d\phi_x$ . The free energy is defined as  $F = -k_B T \ln Z$ .

- We would like to evaluate  $Z$ , and especially *thermal average* of some quantity  $A$ :

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

The dimensionality of the integrals is huge:

$$(\# \text{ points in space}) \times (\text{dim. of field } \phi_x) \sim 10^6 - 10^9.$$

Thus, *some* kind of (quasi?) Monte Carlo integration is required.

- The integral is also *very strongly peaked*:

$$\int [d\phi] \exp \left[ -\frac{H(\phi)}{k_B T} \right] = \int dE n(E) e^{-E/k_B T},$$

which is an integral over sharply localized distribution. Here  $n(E)$  is the density of states at energy  $E$ , i.e.

$$n(E) = \int [d\phi] \delta(H(\phi) - E)$$

Only states with  $E \lesssim \langle E \rangle$  contribute, which is an exponentially small fraction of the whole phase space. This kills standard “simple sampling” Monte Carlo, which is homogeneous over the phase space. This is a form of the so-called **overlap problem** – we are not sampling the interesting configurations nearly often enough.

### 3.2. Importance sampling

Importance sampling takes care of the overlap problem:

- Select  $N$  configurations  $\phi_i$ , chosen with the *Boltzmann probability*

$$p(\phi) = \frac{\exp[-H(\phi)/k_B T]}{Z}$$

Note that in practice we never know  $Z$ ! Thus, we actually do not know the normalization of  $p(\phi)$ . However, this is sufficient for thermal averages.

- Measure observable  $A_i = A(\phi_i)$  from each configuration.
- The expectation value of  $A$  is now

$$\langle A \rangle \equiv \frac{1}{N} \sum_i A_i \rightarrow \langle\langle A \rangle\rangle, \quad \text{as } N \rightarrow \infty. \quad (1)$$

If now the value of  $A_i$ 's are  $\sim$  equal for all  $i$ , **all of the configurations contribute with the same order of magnitude**. This is the central point of importance sampling in Monte Carlo simulations.

The result (1) follows directly from the general results we got in Sect. 1. about importance sampling Monte Carlo integration, or even more directly by realizing that the  $A_i$ -values obviously come from distribution

$$p_A(A') = \int [d\phi] p(\phi) \delta(A' - A[\phi])$$

$$\Rightarrow \bar{A} \equiv \int dA A p_A(A) = \int [d\phi] A[\phi] p(\phi) = \langle\langle A \rangle\rangle.$$

---

***How to generate configurations with  $p(\phi) = \exp[-H(\phi)/k_B T]/Z$ ?***

- Directly – from scratch – we do not know how to do this! (except in some limited simple cases.)
- The basic method (which is really behind almost all of the modern Monte Carlo simulation technology) is based on the proposal by *Metropolis, Rosenbluth<sup>2</sup> and Teller<sup>2</sup>* (J. Chem. Phys 21 1087 (1953)): we can modify existing configurations in small steps, Monte Carlo **update steps**. Thus, we obtain a *Markov chain* of configurations

$$\phi_1 \mapsto \phi_2 \mapsto \phi_3 \mapsto \dots \phi_n \dots$$

With suitably chosen updates and large enough  $n$  the configuration  $\phi_n$  is a “good” sample of the canonical probability  $p(\phi)$ .

How does this work? Let us look at it in detail:

- Let  $f$  be an “update”, a modification of an existing configuration  $\phi$ :  $\phi \xrightarrow{f} \phi'$ . It can be here a “small” update, one spin variable or even a component of it (if  $\phi$  is a vector), or arbitrarily “large” one, for example all of the d.o.f’s (usually just repeated application of small updates).
- **Transition probability**  $W_f(\phi \mapsto \phi')$  is the probability distribution of configurations  $\phi'$ , obtained by one application of  $f$  on some (fixed) configuration  $\phi$ . It naturally has the following properties:

$$W_f(\phi \mapsto \phi') \geq 0, \quad \int d\phi' W_f(\phi \mapsto \phi') = 1$$

The latter property comes from the fact that  $f$  must take  $\phi$  *some-where* with probability 1.

- We can also apply the update  $f$  to probability distributions  $p(\phi)$ :

$$p(\phi) \xrightarrow{f} p'(\phi') \equiv \int d\phi p(\phi) W_f(\phi \mapsto \phi').$$

This follows directly from the fact that we can apply  $f$  to an ensemble of configurations from distribution  $p(\phi)$ .

- What requirements must a “good” update  $f$  fulfill:

A)  $f$  must preserve the equilibrium (Boltzmann) distribution  $p_{\text{eq.}}(\phi) \propto \exp[-H(\phi)/k_B T]$ :

$$\int d\phi W_f(\phi \mapsto \phi') p_{\text{eq.}}(\phi) = p_{\text{eq.}}(\phi')$$

In other words,  $p_{\text{eq.}}(\phi)$  is a *fixed point* of  $f$ .

B)  $f$  must be **ergodic**: starting from *any* configuration, repeated application of  $f$  brings us to arbitrarily close to any other configuration.

- These two simple and and rathe intuitive properties are sufficient to guarantee the following 2 important results:

I) *Any* initial ensemble approaches the equilibrium canonical ensemble  $\rightarrow p_{\text{eq.}}$  as  $f$  is applied to it (“thermalization”).

II) If we sum up the distribution of all of the configurations in a single Markov chain

$$\phi_0 \mapsto \phi_1 \mapsto \phi_2 \mapsto \phi_3 \dots$$

the distribution approaches  $p_{\text{eq.}}$ , as the number of configurations  $\rightarrow \infty$ .

- **Proof:** let  $p(\phi)$  be the probability distribution of the initial ensemble of configurations. After applying  $f$  once to all configs in the ensemble, we obtain the distribution  $p'(\phi')$ . Let us now look how the norm  $\|p - p_{\text{eq.}}\|$  evolves:

$$\|p' - p_{\text{eq.}}\| \equiv \int d\phi' |p'(\phi') - p_{\text{eq.}}(\phi')|$$

$$\begin{aligned}
&= \int d\phi' \left| \int d\phi W_f(\phi \mapsto \phi') (p(\phi) - p_{\text{eq.}}(\phi)) \right| \\
&\leq \int d\phi' \int d\phi W_f(\phi \mapsto \phi') |p(\phi) - p_{\text{eq.}}(\phi)| = \|p - p_{\text{eq.}}\|
\end{aligned}$$

Thus, we get closer to the equilibrium when  $f$  is applied. Nevertheless, this is not sufficient to show that we really get there; it might happen that  $\|p - p_{\text{eq.}}\|$  reaches a plateau without going to zero. This would mean that there exists another fixed point ensemble besides the equilibrium one, i.e. an ensemble which is preserved by  $f$ .

However, the *ergodicity* forbids this: let us assume that  $p$  is another fixed point, and let  $\|p - p_{\text{eq.}}\| > 0$ . Then we can split  $\phi$ 's into two non-empty sets: set  $A$  has those configurations where  $(p(\phi) - p_{\text{eq.}}(\phi)) \geq 0$  and set  $B$  the rest. Because of ergodicity, there must be some configurations  $\phi'$  for which  $W_f(\phi \mapsto \phi')$  is non-zero for some  $\phi$  in set  $A$  and some other in set  $B$ . Thus, on the 3rd line  $\leq \rightarrow <$ , and  $p$  cannot be a fixed point.

---

### 3.3. Detailed balance

The standard update algorithms satisfy the following detailed balance condition:

$$\frac{W_f(\phi \mapsto \phi')}{W_f(\phi' \mapsto \phi)} = \frac{p_{\text{eq.}}(\phi')}{p_{\text{eq.}}(\phi)} = e^{(H(\phi) - H(\phi'))/k_B T}$$

Obviously,  $p_{\text{eq.}}(\phi)$  is a fixed point of this update. Thus, (I) *detailed balance* and (II) *ergodicity* are sufficient for a correct Monte Carlo update. Detailed balance is not a necessary condition; there may be updates which do not satisfy detailed balance. But it is much more difficult then to prove that these updates preserve the Boltzmann distribution. Thus, all of the commonly used update algorithms satisfy detailed balance.

The physical interpretation of detailed balance is *microscopic reversibility* of the update algorithm.

---

### 3.4. Common transition probabilities

The detailed balance condition is a restriction for  $W_f$ ; however, it is still very general. Some of the most common choices for  $W_f$  are (all of these satisfy detailed balance):

- **Metropolis:** (Metropolis *et al*, 1953)

$$W_f(\phi \mapsto \phi') = C \times \begin{cases} \exp[-\delta H/k_B T] & \text{if } \delta H > 0 \\ 1 & \text{otherwise} \end{cases},$$

where  $\delta H \equiv H(\phi') - H(\phi)$  is the change in energy, and  $C$  is a normalization constant.

- **Glauber:**

$$W_f(\phi \mapsto \phi') = \frac{C}{2} \left[ 1 - \tanh \left( \frac{\delta H}{2k_B T} \right) \right] = C \frac{1}{1 + \exp[\delta H/k_B T]}$$

- **Heat bath:**

$$W_f(\phi \mapsto \phi') = C \exp[-H(\phi')/k_B T]$$

This does not depend on old  $\phi$  at all!

- **Overrelaxation:**<sup>1</sup>

$$W_f(\phi \mapsto \phi') = \delta(\phi' - F(\phi)),$$

i.e. the update is **deterministic**:  $\phi \mapsto \phi' = F(\phi)$ . The function  $F$  is chosen so that ( $p = \exp(-H(\phi)/k_B T)$ )

$$\begin{aligned} \text{A) } p(\phi)d\phi &= p(\phi')d\phi' = p(\phi') \left\| \frac{d\phi'}{d\phi} \right\| d\phi \\ &\Rightarrow \left\| \frac{dF(\phi)}{d\phi} \right\| = \exp \left[ \frac{H(F(\phi)) - H(\phi)}{k_B T} \right] \end{aligned}$$

---

<sup>1</sup>The definition of overrelaxation presented here is much more general than the standard definition in literature. However, this has the same generic properties as the standard overrelaxation.

$$\text{B) } F[ F(\phi) ] = \phi.$$

A) means that  $\phi'$  must be as likely to occur as  $\phi$  in thermal equilibrium, and B) implies that applying overrelaxation twice we get back to where we started from. These properties guarantee detailed balance. This update is not generally ergodic! It must be mixed with other updates to achieve ergodicity.

If  $\phi$  is a one-component variable, A) and B) are both met by defining  $\phi' = F(\phi)$  through the cumulants:

$$P(\phi') = 1 - P(\phi), \quad P(\phi) = \int_{\min}^{\phi} d\alpha p(\alpha).$$

The **standard overrelaxation** in the literature is much more intuitive. It assumes that  $H$  is a symmetric function wrt. some reflection in the phase space. For example, there may exist some  $\alpha$  so that

$$H(\alpha - \phi) = H(\phi)$$

for any  $\phi$ . Then the reflection (“overrelaxation”)

$$\phi \mapsto \phi' = F(\phi) = \alpha - \phi$$

satisfies A) and B) and is a good (and fast!) update.

---

- Note that, in principle, the update  $\phi \mapsto \phi'$  above can refer to any size of update; for example, we might update spins at each location on a lattice  $\phi_x \rightarrow \phi_x + C(X_x - 0.5)$  and do a global accept/reject step with Metropolis  $W_M$ . However, in this case  $C$  should be very small to achieve good acceptance rate.
- Global heat bath update  $\Leftrightarrow$  generate a whole new configuration from scratch! Normally impossible.
- Standard (and easiest) way is to update the spins separately at each location.

- Note that  $W_f$  is really a probability density. Thus, the new configuration  $\phi'$  must be chosen with care to ensure correct distribution (for example, choose it with appropriate measure, and accept/reject it with  $W_f$ . This is not usually optimal, though!)
- 

### 3.5. Updating $O(2)$ sigma model:

- Let us illustrate these update methods with a concrete example,  $O(2)$   $\sigma$ -model or the XY model:

$$H/k_B T \equiv S = -\beta \sum_{\langle xy \rangle} s_x \cdot s_y = -\beta \sum_{\langle xy \rangle} \cos(\theta_x - \theta_y)$$
$$Z = \int_{-\pi}^{\pi} \left[ \prod_x d\theta_x \right] e^{-S[\theta]}$$

Here  $s_x$  is a 2-component vector with  $|s_x| = 1$ , and  $\theta_x$  is its angle from, say, 1-axis.  $x$  and  $y$  are discrete coordinate vectors (in 2 or

3 dimensions), and  $\langle x, y \rangle$  refers to *nearest-neighbour* coordinate pairs.

---

- **Physics of the model:** In 3 dimensions the XY model has a phase transition at  $\beta = \beta_c = 0.454165(4)$ . The model exhibits *spontaneous symmetry breaking*; there is spontaneous magnetization if  $\beta > \beta_c$  ( $T < T_c$ ), i.e.

$$\langle |M| \rangle = \frac{1}{V} \left\langle \left| \sum_x s_x \right| \right\rangle \neq 0 \quad \text{as } V \equiv N^3 \rightarrow \infty.$$

The transition is of second order, and the most important critical exponents have been measured/calculated; for example,

$$|M| = (\beta - \beta_c)^b \quad b \approx 0.35$$

**Universality:** Phase transitions in systems which have 2d (internal) rotational symmetry (O(2)) have the same critical exponents as the XY model:

- superfluidity  $\lambda$ -transition in liquid  $^4\text{He}$  (space shuttle experiment)
- ferromagnets with an “easy plane”
- some liquid crystals
- density waves
- type II superconductors (?)

In 2 dimensions, there is no magnetization,<sup>2</sup> but there is still a phase transition, *Kosterlitz-Thouless* transition.

A lot more information about XY model can be found in [Pelissetto, Vicari, cond-mat/0012164], for example.

---

---

<sup>2</sup>This is due to the *Coleman-Mermin-Wagner theorem*: in 2d, continuous symmetries cannot break spontaneously!

- Choose a variable at site  $x$  for updating
- Calculate the *local action*, the part of the action which depends on the variable  $s_x$ :

$$S_x(s_x) = -\beta \sum_{y=\text{n.n. of } x} s_x \cdot s_y = -s_x \cdot V = -v \cos \alpha$$

Here  $v = |V|$  and  $\alpha$  is the angle between  $s_x$  and  $V$ . When we modify the variable  $s_x$ , the change in total action is  $\delta S = \delta S_x$ .

- **Heat bath:**

Choose new  $s_x$  with probability

$$p(s_x) \propto e^{-S_x(s_x)} = e^{v \cos \alpha}$$

- Satisfies detailed balance

- Computer gives random numbers from uniform distribution,  $X \in [0, 1)$ . We obtain  $s$  from distribution  $p(s)$  from the inversion

$$X = \int_{\min}^s ds' p(s') = C_\alpha \int_{-\pi}^{\alpha} d\alpha' e^{v \cos \alpha'}$$

where  $C_\alpha = \int_{-\pi}^{\pi} d\alpha' e^{v \cos \alpha'}$  is a normalization constant.

- We're unlucky:  $p(s)$  is not very easily integrable. See sect. 2., where we discussed how to do generate random numbers just from this kind of distribution.

- Integrability is often a problem for many actions. For an easier case, consider O(3) sigma model. However, we can always generate random numbers with distribution  $p(s)$  by using some version of the rejection method.

- Heat bath update is quite common in Monte Carlo simulations. In gauge theories, efficient implementation  $\exists$  for SU(2)

- **Metropolis:**

How to generate the new angle variable with the Metropolis probability form? Integrating  $W_M$  is as complicated as the heat bath integral. However, we may do an efficient *restricted* Metropolis transition by using the following accept/reject method:

Choose new  $\theta'_x$  with

$$\theta'_x = \theta_x + C(X - 0.5) \quad (\text{mod } 2\pi),$$

where  $C$  is a tunable constant.  $\theta'_x$  is *accepted* with probability

$$W_M(\theta_x \mapsto \theta'_x) = \min(1, e^{S_x(\theta_x) - S_x(\theta'_x)}),$$

if rejected, either repeat or leave  $\theta_x$  as is.

- Satisfies detailed balance.

- More generally: instead of using the uniform distribution  $\theta'_x \in [\theta_x - C/2, \theta_x + C/2]$ , we could choose  $\theta'_x$  from any distribution  $f(\theta_x - \theta'_x)$ , which satisfies  $f(a) = f(-a)$ . For example,  $f(a)$  could be a Gaussian distribution.

- Function  $f$  (or constant  $C$ ) is tuned to optimize the performance; usually so that the acceptance is  $\sim 50 - 70\%$ .

- If  $C = 2\pi$ ,  $\theta'$  will be evenly distributed from 0 to  $2\pi$ . However, the rejection rate will be (usually) very large.

- Metropolis is very versatile method, and can be applied to almost any problem. We only have to evaluate  $e^S$ ! No integrals or inversions. One Metropolis update is also usually quite fast. It is also the “original” update, first one to be used in a real simulation.

- If you repeat Metropolis update of  $\phi_x$  (fixed site) many times, the final distribution  $\mapsto$  heat bath update!

- **Overrelaxation:**

Reflect  $s_x$  to the “other side” of the potential  $S_x$ :

$\alpha \rightarrow -\alpha$ , or equivalently

$$s_x \rightarrow 2 \frac{s_x \cdot V}{v^2} - s_x.$$

- Deterministic, very fast to implement.

- Satisfies detailed balance

- *Not ergodic*:  $S$  never changes: in other words, the update is *microcanonical*.

- Must be usually mixed with other updates to achieve ergodicity.
  - Nevertheless, often overrelaxation is more efficient than the other updates above.
  - In this case  $S_x(s)$  is always symmetric wrt.  $V$  ( $V$  varies from site to site and update to update, but it is always simple to find!). If  $S_x$  is not symmetric, overrelaxation is much more difficult (often not worth it).
  - Not applicable to discrete spins.
- The Glauber form (and also other transition probabilities which depend on  $\delta S_i$ ) can be implemented very much along the lines of Metropolis. However, acceptance becomes slightly worse than in Metropolis → possibly worse performance.
-

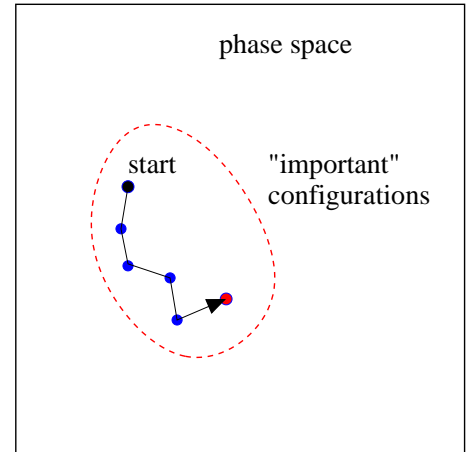
Repeat the above single-site ( $\phi_x$ ) updates for all of the sites on the lattice  
→ update sweep.

Typical program structure:

- 1 or more update sweeps
- measure what you want, accumulate or print
- repeat until enough statistics

Even after all of the sites have been updated, the system evolves only a finite step in the phase space; it “remembers” the previous configuration.

**Autocorrelation time  $\tau_A$ :** number of update sweeps required to make the configuration statistically independent from the starting configuration.



⇒ The number of independent configurations in a Monte Carlo simulation of  $N$  configurations is  $N/\tau_A$ .

⇒ **Errors**  $\propto \sqrt{\tau_A/N}$ .

Thus, a good update minimises  $\tau_A$ . Rule of thumb:

**Metropolis < heat bath < overrelaxation**

where < shows the “goodness” relation of the algorithm, i.e.  $1/\tau_A$ . However, the real performance depends on the details. We always should compare to the (wall-clock) time, not to the # of updates!

Because of non-ergodicity, overrelaxation is usually mixed with heat bath or Metropolis: for example, we can do 5 overrelaxation sweeps + 1 heat bath sweep.

Autocorrelation time diverges at phase transition points (or more generally, when the correlation length diverges): *critical slowing down*.

### 3.6. Ising model

The Ising model is the simplest discrete spin model. In this case the field variable  $\phi_x \rightarrow s_x = \pm 1$ , and the partition function is

$$Z = \sum_{\{s_x = \pm 1\}} \exp \left[ -\beta \left( \frac{1}{2} \sum_{\langle xy \rangle} (1 - s_x s_y) + H \sum_x s_x \right) \right]$$

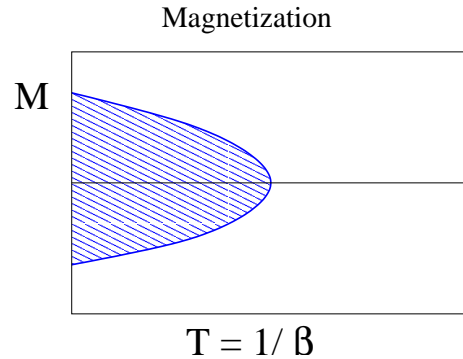
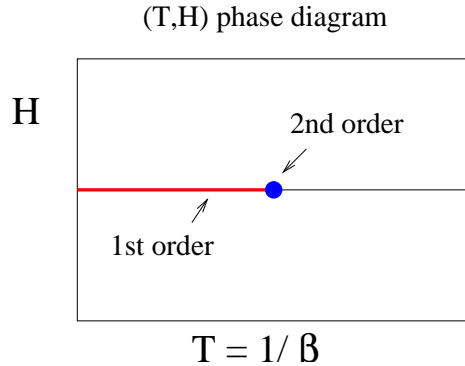
Here  $\langle xy \rangle$  goes over the nearest-neighbour sites of the rectangular lattice, and  $H$  is an external magnetic field. We shall consider here mostly the case where  $H = 0$ .

When  $H = 0$  the Ising model has a second-order phase transition in all dimensions  $> 1$ . In 2 dimensions, the Ising model has been solved analytically (Onsager), and the transition is at  $\beta_c = \ln(1 + \sqrt{2}) \approx 0.88137$ . In 3 dimensions,  $\beta_c \approx ???$

If  $\beta > \beta_c$  (" $T < T_c$ "), the system shows spontaneous magnetization:

$$\langle |M| \rangle = \left\langle \left| \frac{1}{V} \sum_x s_x \right| \right\rangle \neq 0$$

## Phase diagram:



## Universality:

- Ferromagnetism, Curie point
- Liquid-vapour transition (lattice gas)
- Generic transition for systems without explicit symmetry

### 3.6.1. Updating the Ising model

- Let us assume here  $H = 0$ , and that we are in 2 dimensions.
- Choose spin  $s_x$ . The *local energy functional*

$$S_x(s_x) = \beta/2 \sum_{y=\text{n.n. of } x} (1 - s_x s_y) = \beta/2 \left( 4 - s_x \sum_{y=\text{n.n. of } x} s_y \right)$$

$S_x$  has only 5 possible values! (Can be calculated beforehand.)

- **Heat bath:** Choose new  $s_x = \pm 1$  with probability

$$p(s_x) = \frac{e^{-S_x(s_x)}}{e^{-S_x(+1)} + e^{-S_x(-1)}}$$

(see sect. 2.)

- **Metropolis:** Flip the spin  $s_x \mapsto s'_x = -s_x$ , and calculate  $\delta S_x = S_x(s'_x) - S_x(s_x)$ . Accept this flip with probability

$$p_{\text{accept}} = \min(1, e^{-\delta S_x})$$

If the update is rejected, leave  $s_x$  as is (do not redo until accepted!  
You can redo Metropolis a fixed number of times, though.)

- **Glauber:** Substitute  $p_{\text{accept}} = 1/(1 + e^{\delta S_x})$  above.
- **Kawasaki:** Choose a nearest-neighbour pair  $s_x, s_y$ . If  $s_x \neq s_y$ , consider exchanging the spins  $s_x \mapsto s'_x = s_y$  and  $s_y \mapsto s_x$ . This is accepted with the Metropolis probability

$$p_{\text{accept}} = \min(1, e^{-\delta S})$$

If the update is rejected, leave the pair as is.

Kawasaki update preserves the magnetization  $M$ . It is used if we really want to study an ensemble with fixed magnetization (lattice gas: fixed particle number!)

### 3.7. Boundary conditions

How do we choose the boundary conditions on a rectangular lattice? There are several choices, usually dictated by the physics we want to study. Let us assume 2-dimensional  $N \times N$  lattice, with coordinates  $x, y = 1 \dots N$ .

- **Periodic boundaries:** This is the most popular choice when we want to *minimize* the effect of boundaries. The neighbours wrap around the edges, thus the coordinate pairs  $(N, y) \leftrightarrow (1, y)$  and  $(x, N) \leftrightarrow (x, 1)$  are neighbours.

Topology: 2-dim. torus (surface of a donut)

The boundaries are not special points, the system has translational invariance!

- **Fixed boundaries:** The boundary layers ( $x = 1, N$  or  $y = 1, N$ ) are fixed either all to the same value or to some specially chosen values. Strong boundary effects.

- **Free boundaries:** The boundary sites just have less neighbour sites. Strong boundary effects.
- **Twisted periodic boundaries:** For example, the spins  $s(N, y)$  and  $s(1, y)$  are neighbours, but with “inverted” sign – i.e. the action has opposite sign for these particular links. When  $\beta > \beta_c$  the system has magnetization, but because of the “twist” there must be an interface (kink) somewhere in the system. Despite the appearance, this boundary is also translationally invariant: the boundary does not form a special point.

This is used to study the properties of the interface between states with +1 and -1 magnetization.

### 3.8. Structure of the Monte Carlo program

**Traversal order:** in which order should we go through the points on the lattice? In principle this is (largely) up to you, except in some special cases. The common orders are:

- **Typewriter ordering:** go through the sites row-by-row, from left to right. Fast & recommended, for standard uses.

However: breaks detailed balance for Ising model + Metropolis update! (at least in 1 dimensions... )!!

- **Random ordering:** pick the site at random. Good, but in practice computationally slower than typewriter. Used in some *real-time* calculations.
- **Checkerboard ordering:** divide the sites into black and white sets ( $x + y$  even/odd), as in the checkerboard. Since each black site has only white neighbours and vice versa, we can update all of the black

sites independently from each other while keeping white sites fixed.  
Repeat for white sites.

This must be used in vector computers (Crays, many other older supercomputers) and in parallel programming.

---

## Sample: heat bath for Ising

```
#define NX 32
#define NY 32
...

int s[NX][NY];
int sum,x,y;
double beta,p_plus,p_minus;
...

/* sum over the neighbour sites - typewriter fashion */
for (x=0; x<NX; x++) for (y=0; y<NY; y++) {
    sum = s[xup[x]][y] + s[xdn[x]][y]
        + s[x][yup[y]] + s[x][ydn[y]];

    /* Heat bath update - calculate probability of +-1 */
    p_plus = exp( (beta/2.0) * sum ); /* prob. of +1, unnormalized */
    p_minus = 1.0/p_plus; /* and -1 */
    p_plus = p_plus/(p_plus + p_minus); /* normalized prob of +1 */

    /* and choose the state appropriately */
    if (mersenne() < p_plus) s[x][y] = 1; else s[x][y] = -1;
}
```

### 3.9. *Some implementation details*

#### 1. **Looping over the lattice:**

2 common ways to organize the lattice arrays:

##### **A) 1 variable/dimension:**

```
int s[NX][NY];
```

In this case one should loop over  $y$ -coordinate in the *inner* loop:

```
for (x=0; x<NX; x++) for (y=0; y<NY; y++)
```

in C, the last index is the “fastest,” i.e. these variables are stored in consecutive locations.

In Fortran, this is opposite:

```
integer s(NX,NY)
```

Here the first index ( $x$ ) is faster.

The speed difference varies a lot depending on the problem/computer.

## B) 1 loop variable only:

```
#define VOLUME (NX*NY)
int s[VOLUME];
```

Looping:

```
for (i=0; i<VOLUME; i++) s[i] ...
```

Used a lot in old vector supercomputers – long loops, vectorizes effectively.

---

## II. Fetching the neighbours:

Typically (simple) MC programs use a significant amount of time fetching the neighbours of a site (periodic boundaries!). There are several ways to do this:

A) Use modulus to get the neighbours:

$$x_{\text{up}} = (x + 1) \bmod NX, x_{\text{down}} = (x + NX - 1) \bmod NX$$

- works in C [ $x = 0 + \dots (NX - 1)$ ], has to be modified a bit in Fortran.
- slow, *unless*  $NX = 2^n$  and  $NX$  constant (so that the compiler knows what it is).

B) Tabulate the neighbours beforehand:

```
for (x=0; x<NX; x++) {  
    xup[x] = (x+1) % NX;  
    xdn[x] = (x-1+NX) % NX;  
}
```

(same for y-coordinate) and use the tables `xup[ ]` etc. in the loop. This was used in the example.

- Usually pretty good. If the size is always the same to x- and y-directions, then single up,down -arrays can be used.
- Has to be modified a bit in Fortran:

```
xup(x) = mod(x, NX) + 1
```

etc.

C) If single volume index (I.B) is used, then it is usually easiest to use global neighbour pointer arrays, which are again initialized at the beginning:

```
#define ixy(x,y)  ((x+NX)%NX + NX*((y+NY)%NY))
...
for (i=0; i<VOLUME; i++) {
    x = i % NX;
    y = i / NX;
    xup[i] = ixy(x+1,y);
    xdn[i] = ixy(x-1,y);
    yup[i] = ixy(x,y+1);
    ydn[i] = ixy(x,y-1);
}
```

Here `xup[ ]` etc. is an integer array of size `VOLUME`.

– Easy to use in the loops over volume. However, neighbour arrays are large, and memory access can become expensive.

– Again, this style vectorizes easily. It is also used in parallel programming.

---

### III. Variable type:

For discrete models (like Ising), it is usually worthwhile to use the smallest easy variable (in this case, (unsigned) `char`).

However, gains depend very much on details.

---

**IV. Structure of the program:** The program flow of a Monte Carlo simulation program is usually as follows:

1. Initialize what is needed:
  - seed the random numbers
  - initialize the configuration, or
  - load a stored configuration
  - initialize neighbour arrays etc.
2. Update, i.e. do one or more update sweeps through the system. Different algorithms can be used.
3. Measure, and either
  - accumulate the results or
  - write results to a file for post-processing (preferred).
4. Return to 2, until we have the desired amount of measurements.
5. Do whatever maintenance is needed (for example, save the configuration)

### 3.10. Measurements

The configurations of the model are generated with some algorithm, such as Metropolis. We want to measure numerically thermodynamic quantities of interest, for example (for Ising model)

- Energy density  $E = -\frac{1}{V} \sum_{\langle i,j \rangle} s_i s_j$
- Magnetization  $M = \frac{1}{V} \sum_i s_i$
- Correlation function  $\Gamma(z) = \frac{1}{V} \sum_i s_i s_{i+z}$
- Correlation length  $\xi$ :  $\Gamma(z) \sim e^{-z/\xi}$
- Specific heat:  $C_V = \frac{1}{V} \frac{\partial E}{\partial T} = \langle E^2 \rangle - \langle E \rangle^2$
- Magnetic susceptibility:  $\chi_M = \frac{1}{V} \frac{\partial M}{\partial T} = \langle M^2 \rangle - \langle M \rangle^2$

Note that the last 2 measurements do not require “new” measurements; these can be calculated directly from measurements  $E_i$ ,  $M_i$  (if these have been stored in a file, for example).

The correlation function requires usually special methodology.

---

### *3.11. Phase transitions and critical exponents*

Most phase transitions are described by an **order parameter** which is zero in one phase (disordered phase), non-zero in the other phase (ordered phase). Thus, it cannot be an analytic function at the critical point.

- First order — the order parameter (and in general almost any thermodynamical quantity) has a discontinuous jump; i.e. the 1st derivative of the partition function.
  - latent heat, discontinuity in energy
- Second order — the susceptibility or specific heat are divergent (in general, second derivatives of partition function).

Second order transitions are classified by their **critical exponents**, which measure the divergence at the critical point:

$$M \sim |T - T_c|^\beta$$

$$\chi_M \sim |T - T_c|^{-\gamma}$$

$$C_V \sim |T - T_c|^{-\alpha}$$

$$\xi \sim |T - T_c|^{-\nu}$$

For the 2d Ising model, these exponents are  $\alpha = 0$ ,  $\beta = 0.125$ ,  $\gamma = 1.75$ ,  $\nu = 1$ .

However, on a finite lattice we have finite number of degrees of freedom and *everything* is analytic! Thus, on a finite lattice the order parameter is either always non-zero or always zero. Indeed:

$$M = \left\langle \frac{1}{V} \sum_i s_i \right\rangle \equiv 0 \qquad |M| = \left\langle \left| \frac{1}{V} \sum_i s_i \right| \right\rangle > 0$$

always on a finite lattice! Careful **finite size analysis (FSS)** is needed.  
(Return to that later)

### 3.12. Autocorrelations

In sect. 3.5. we already mentioned that the Monte Carlo simulations suffer from autocorrelations: since the update step is a smaller or larger modification of some configuration (Markov chain!), successive configurations and measurements are correlated.

- Let  $X_i$  be the measurements of some quantity  $X$  from configuration number  $i = 1 \dots N$ . At finite  $N$ , the error is

$$\delta X = \sqrt{\frac{\sum_i (X_i - \langle X \rangle)^2}{N(N-1)}}$$

if and only if the measurements  $X_i$  are statistically independent.

- We can define an **autocorrelation function** of quantity  $X$ :

$$C(t) = \frac{\frac{1}{N-t} \sum_{i=1}^{N-t} X_i X_{i+t} - \langle X \rangle^2}{\langle X^2 \rangle - \langle X \rangle^2} \sim e^{-t/\tau_{\text{exp}}}$$

where the last point holds if  $N \rightarrow \infty$  and  $t \rightarrow \infty, t \ll N$ .

- The denominator is there to normalize  $C(0) = 1$ .
- $\tau_{\text{exp}}$  is the exponential autocorrelation time. This is in principle (almost) unique; i.e. almost all observables show the unique longest autocorrelation time, which really measures when the configurations become thoroughly uncorrelated.
- However, for error analysis, the relevant quantity is the **integrated autocorrelation time**:

$$\tau_{\text{int}} = \frac{1}{2} + \sum_{t=1}^{\infty} C(t)$$

Note that  $\tau_{\text{int}} \approx \tau_{\text{exp}}$  if the autocorrelation function is purely exponential,  $C(t) \approx e^{-t/\tau_{\text{exp}}}$ . However, usually  $\tau_{\text{int}} < \tau_{\text{exp}}$ .

- In Monte Carlo analysis with correlated measurements, the error estimate is

$$\text{error of } X \equiv \delta_X = \sqrt{2\tau_{\text{int}}} \delta_{x,\text{naive}} = \sqrt{2\tau_{\text{int}} \frac{\sum_i (X_i - \langle X \rangle)^2}{N(N-1)}}$$

Here  $\delta_{X,\text{naive}}$  is the naive error shown on previous page.

- How to calculate  $\tau_{\text{int}}$ ? One has to write all measurements in a file during the simulation. However, the double summation can become expensive, and because  $C(t)$  is *noisy when*  $t \gg \tau_{\text{int}}$ , the sum in  $\tau_{\text{int}}$  can behave badly when  $t$  is large. Thus, the sum can be cut self-consistently to values, for example,  $t \leq 6\tau_{\text{int}}$ , as the summation proceeds.
-

### 3.13. Error estimates in Monte Carlo measurements

There are 2 commonly used methods for estimating the errors in Monte Carlo measurements:

#### 1. Use

$$\delta_X = \sqrt{2\tau_{\text{int}}} \delta_{x,\text{naive}}$$

as above.

- #### 2. Block the measurements in $M$ bins of length $m$ . Calculate averages of observables in each bin, $X_k^b$ , $k = 1 \dots M$ . If the bin length $m \gg \tau_{\text{int}}$ , the bins are statistically independent, and we can use the naive formula

$$\delta X = \sqrt{\frac{\sum_{i=1}^M (X_i^b - \langle X \rangle)^2}{M(M-1)}}$$

In practice, one divides the set in variable number of blocks. The error estimate should be  $\sim$  constant if the bin length is large enough (sometimes one has to extrapolate to block length  $\rightarrow \infty$ ).

Note that neither the autocorrelation nor the blocking method change the expectation value  $\langle X \rangle$  in any way! Just the error bars.

Clearly, in order to minimize the errors, we want to use an update which has as small  $\tau$  as possible – but measured in CPU-time used, not in iterations!

If we manage to improve  $\tau \mapsto \tau/10$ , then either

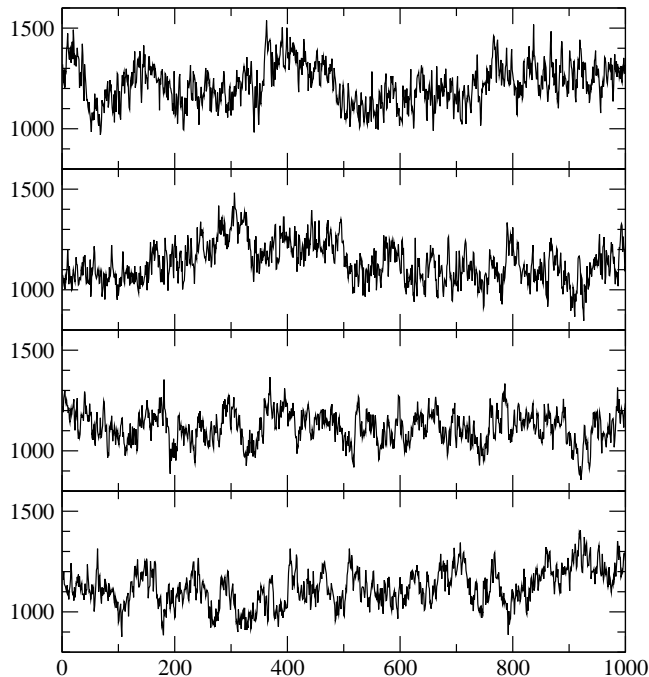
- we need a factor of 10 less configurations for given accuracy
  - for a fixed number of configurations, the errors are reduced by a factor of  $\sqrt{10} \sim 3$ .
-

### 3.14. Example: Ising model

Time histories of the measurements of the total energy, measured from  $64^2$  Ising model at  $\beta_c$ . Sample of 1000 (from a total of 400000 each).

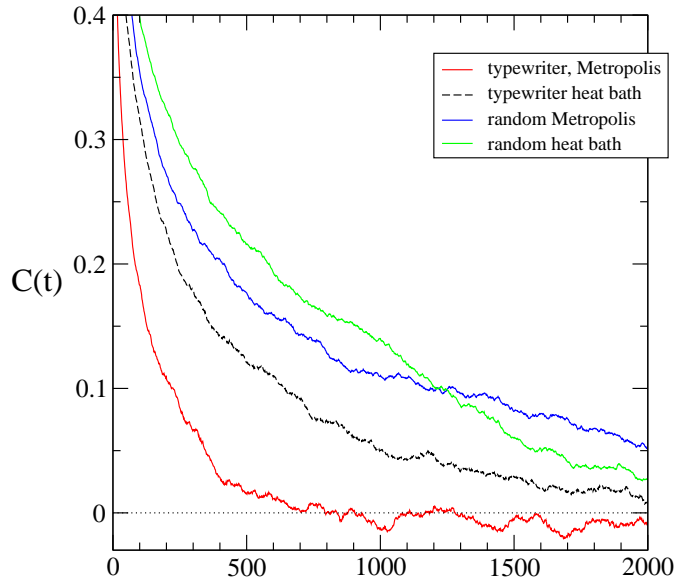
From top to bottom: Metropolis and heat bath with “typewriter” ordering, then Metropolis and heat bath with random ordering.

It is next to impossible to see by eye which of these algorithms has the shortest autocorrelations.



The autocorrelation function of the total energy from the previous case. Clearly, the Metropolis update with typewriter ordering is the fastest of these algorithms.

However, this is characteristic for the Ising model - in general heat bath tends to be faster than Metropolis!



The *integrated* autocorrelation time  $\tau_{\text{int}}$ , measurement average, and error calculated in various ways are as follows (with 400000 update+measurement cycles for each case):

	$\tau_{\text{int}}$	$\langle E \rangle$	$\delta_{\text{naive}}$	$\delta_{\tau}$	$\delta_{100}$	$\delta_{1000}$	$\delta_{10000}$
Metro, typew.	54	1182.44	0.17	1.78	1.03	1.76	2.18
HB, typew.	157	1176.45	0.17	3.01	1.18	2.50	3.04
Metro, random	271	1181.29	0.17	3.99	1.24	2.80	3.52
HB, random	316	1180.66	0.17	4.26	1.26	2.97	4.34

The Metropolis with typewriter ordering is the best of the bunch - for fixed # of updates. The quantity  $\delta_{100}$  means the error using binned naive estimate, the number is the bin size. Clearly, the bin length has to be  $\gg \tau$  in order for the method to work!

The real figure of merit is obtained when we compare the time used to the number of configurations:

	time(ms)/iteration	time(ms)/iter. $\times 2\tau_{\text{int}}$
Metro, typew.	1.0	108
HB, typew.	1.2	364
Metro, random	1.4	748
HB, random	1.6	1004

Here the times are in milliseconds. The last column is the real measure of the efficiency of the algorithm (and implementation): how much time is needed to obtain one *independent* configuration ( $t \sim 2\tau$ ).

However, the random ordering is still useful: using random ordering the evolution of the system can be in some cases interpreted as a **real-time** evolution ( $\sim$  Glauber dynamics).

### 3.15. Thermalization

Because of autocorrelations, and because we typically start the simulation from a “bad” configuration (often fully ordered, “cold”, or disordered, “hot” configuration), the initial measurements are just wrong. We should get rid of these.

One should discard at least  $n \gg \tau$  measurements from the beginning, often values

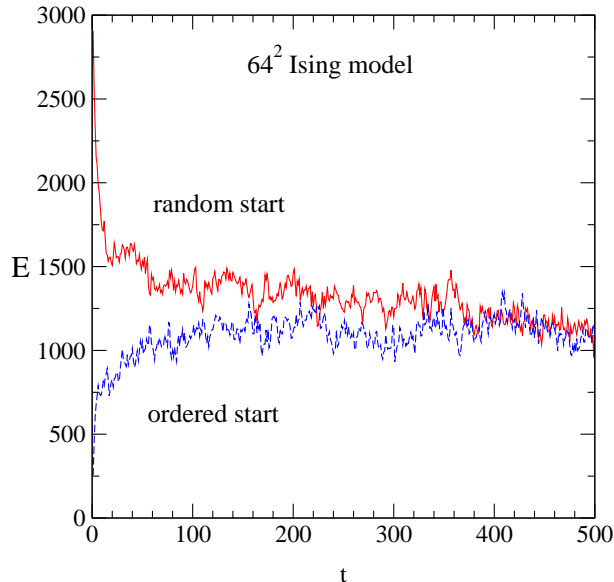
$$n = 5 \dots 10 \tau$$

are used. This depends on the problem, though.

Autocorrelation time  $\tau$  can be very large – if the thermalization time  $\lesssim \tau$  and measurement time  $\sim \tau$ , the results are very much suspect! This can happen in practice in Monte Carlo calculations, because of **critical slowing down**.

In the binning method of error analysis (previous subsection), the data from the initial bin(s) should be discarded.

Hot and cold starts are also often used to estimate the needed thermalization:



This is often actually used to investigate metastable states – for example, in 1st order phase transitions.

### 3.16. Critical slowing down

- How does the statistical error (and hence, the cost in cpu-time) behave in Monte Carlo simulations?
- We have already established ( $N$  = number of sweeps)

$$\delta \sim \frac{1}{(\# \text{ indep. configs})^{1/2}} \sim \sqrt{\tau_{\text{int}}/N}.$$

- More precisely, when we consider also the system size, the error behaves as

$$\delta \sim (\tau_{\text{int}}/N)^{1/2} (\xi/L)^{d/2}$$

where  $d$  is the dimensionality of the system,  $\xi$  is the correlation length and  $L$  the *linear* system size (volume =  $L^d$ ).

- Why  $(\xi/L)^{d/2}$ ? Now  $(L/\xi)^d = \#$  of independent  $\xi$ -sized domains in the volume. These are, in practice, statistically independent, so that the total # of independent correlation volumes =  $N/2\tau \times (L/\xi)^d$ .

- The autocorrelation time  $\tau \sim \xi^z$ , where  $z$  is a dynamical exponent which depends on the update algorithm.
- How about the cpu-time cost? Naturally, the time for one sweep is  $\propto L^d$ . Thus, if we require errors of some definite level  $\delta$ , the time behaves as

$$\text{time} \sim \tau_{\text{int}} \xi^d / \delta^2$$

Surprisingly, if  $\xi$  is constant (in lattice units), this is independent of  $L$ ! Because large volumes help to get rid of finite-size effects, it pays to use as large volumes as practically possible.

---

- However, the situation is very different when we are at or near a critical point (or continuum limit). In this case  $\xi$  diverges, and, in principle, the autocorrelation time

$$\tau \sim \xi^z \sim |T - T_c|^{-z\nu}$$

diverges also at the critical point! This is known as the *critical slowing down*.

- However, on a finite volume, the correlation length is cut-off by the system size  $\xi \sim L$  — or, in other words, when the action has no built-in mass/length scale, the scale is set by the infrared cutoff — the system size.
- Thus, at the critical point,

$$\tau \sim L^z.$$

The exponent  $z$  is called the *dynamical critical exponent*. The cost of the simulation now behaves as

$$\text{cost} \sim L^{d+z}.$$

instead of being  $\sim \text{constant}$ !<sup>3</sup> Here  $L^d$  just reflects the fact that

---

<sup>3</sup>In reality, the cost depends quite a lot on the observable: some observables have non-trivial volume dependence at a critical point.

the correlation volume  $\sim$  total volume, due to the physics. This we cannot get rid of. However, different update algorithms have different values of  $z$ .

- Because interesting physics happens at the phase transition points, and because obtaining meaningful results at a critical point requires finite-size scaling (FSS) scaling analysis, it is important to use algorithms with as small  $z$  as possible.

- 
- Common stochastic algorithms — heat bath, Metropolis — have  $z \approx 2$ : the nearest-neighbour update is a stochastic process, so that the signal propagates over a distance  $\xi$  in time  $\propto \xi^2$  (diffusion, random walk).

More precisely, it has been argued that for stochastic algorithms  $z \geq \gamma/\nu$ .

- Some **non-stochastic** update algorithms may have  $z = 1$ , or the sig-

nal propagates over a distance  $\xi$  in time  $\propto \xi$ . Physically, this means that the update algorithms support some kind of **wave motion** instead of diffusion.

Examples: Overrelaxation in many cases has  $z \approx 1$ ; also evolving the lattice fields using equations of motion. However, if the process involves randomness,  $z \approx 2$  at the limit  $L \rightarrow \infty$ .

Thus, for many practical purposes, a good update is a mixture of overrelaxation and Metropolis/heat bath.

- $z \ll 1$ , and even  $z = 0$  can be achieved in some cases using **non-local update algorithms** (cluster, multigrid ...).

*Cluster algorithms* are the most succesful non-local update methods.