10 Multicanonical methods

Normal Monte Carlo algorithms sample configurations with the Boltzmann weight $p \propto \exp(-\beta E)$. Sometimes this is not desirable.

> Example: if a system has a first order phase transitions between two states the system is preferably in either of the states (at the transition temperature). The tunnelling between the states can be strongly suppressed. If the goal is to study the tunnelling process, the Monte Carlo simulation should enhance the probability of those configurations!

Multicanonical methods tailor the MC probability in order to enhance configurations in desired phase space domains.

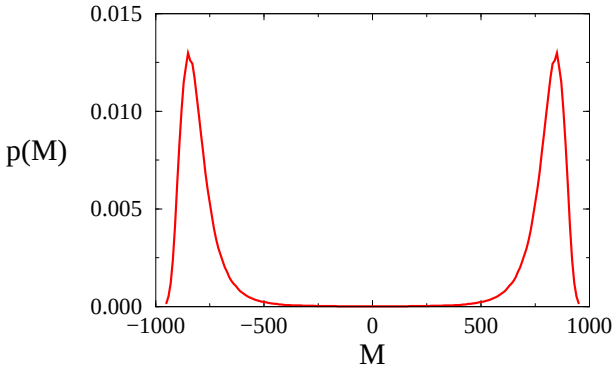
Related methods:

- Umbrella sampling [Torrie, Valleau, J.Comp.Phys. 23 (1977)]
- Multicanonical [Berg, Neuhaus, Phys.Lett.B 267 (1991)]
- Simulated tempering [Marinari, Parisi, Europhys.Lett. 19 (1992)]
- Method of expanded ensembles [Lyubartsev et al, J.Chem.Phys. 96 (1992)]
- Parallel tempering [Hukushima, Nemoto 96]
  [Geyer 91]
- ...

[review: B. Berg, cond-mat/9909236]

10.1 1st order phase transitions

Consider again Ising model, this time at $T < T_c$: now it has a 1st order phase transition between states with positive and negative magnetization.
Magnetization distribution $p(M)$, volume 32^2, $\beta = 1/T = 0.453$, $h = 0$:

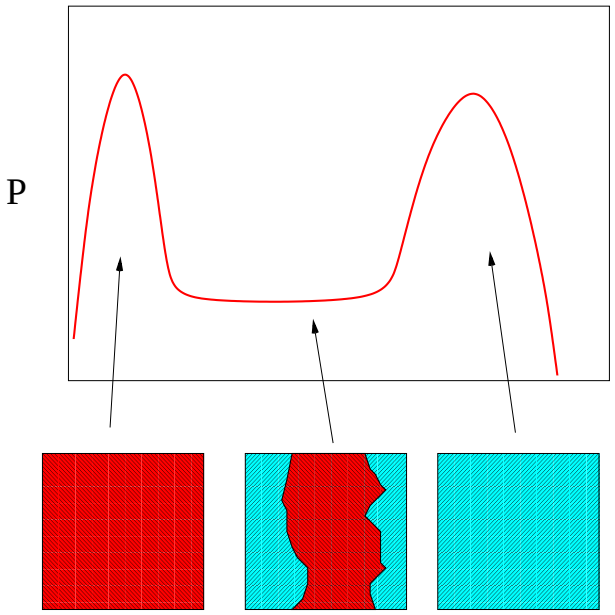


Canonical probability

$$p_{\rm can}(M) \propto \sum_{\{s\}} \delta(M, M_s) e^{-\beta E_s}$$

is strongly suppressed between the peaks. → canonical MC simulations hardly ever sample those configurations!

Why is the probability suppressed? This is due to the fact that the system at low temperatures wants to remain ordered, either to positive or negative magnetization. In order to get to the "middle", the system has to form a state which is "mixed": they contain a fraction of both pure phases, with are separated by **interfaces**.



The interface carries extra free energy, $f_A = \sigma A$, where σ = **surface tension**, and A is the area of the interface. Minimally $A = 2L$ on 2-dim. lattice with periodic boundaries. This causes *exponential suppression* of the configurations in the middle:

$$p_{\rm min}/p_{\rm max} \sim e^{-\sigma 2L}$$

This can actually be used to measure the surface tension σ, and it is one of the major uses for the multicanonical method.

10.2 Multicanonical method:

- Instead of using the Bolzmann weight $p \propto \exp[-\beta E]$ in Monte Carlo simulations, use a modified probability

  $$p \propto \exp[-\beta E + W(M)]\,,$$

  where $W(M)$ is a weight function, inserted 'by hand', which is carfully tuned to enhance the probability between the peaks.

- Multicanonical probability is produced by multicanonical simulations:

  $$p_{\rm muca}(M) \propto \sum_{\{s\}} \delta(M, M_s) e^{-\beta E_s + W(M_s)} \propto p_{\rm can}(M) e^{W(M)}$$

- If we want that $p_{\rm muca}(M)$ is flat in the region between the peaks we should choose $W(M) = -\log p_{\rm can}(M)$. However, we do not a priori know $p_{\rm can}(M)$. Thus, one has to do with approximations (later).

- Multicanonical simulation gives us $p_{\rm muca}(M)$. We obtain the canonical distribution (which is, after all, what we're after!) by reweighting:

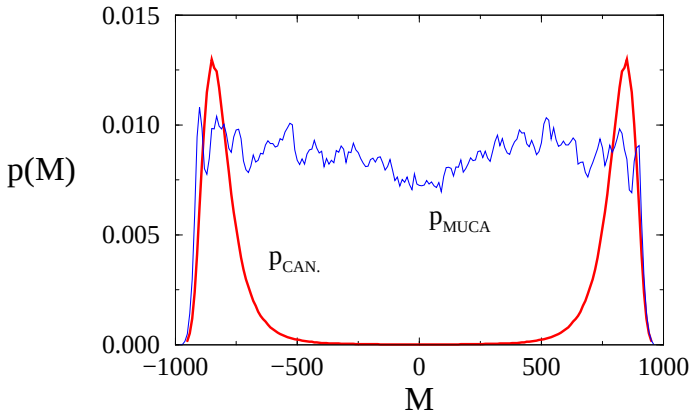
  $$p_{\rm can}(M) \propto p_{\rm muca}(M) e^{-W(M)}$$

- Likewise, we can also reweight the measurements of any observable from multicanonical simulation to canonical one: let us consider measurements O_i of observable O, measured from configurations obtained from multicanonical simulation. Then we can obtain the canonical expectation value of O by reweighting

  $$\langle O \rangle_{\rm can} = \frac{\sum_i O_i e^{-W(M_i)}}{\sum_i e^{-W(M_i)}}\,.$$

  Thus, this is just more general form of the reweighting presented earlier (there $W(M) = \delta h M$)!

- The picture below shows $p_{\rm muca}(M)$, which is $\sim$ constant. The wigglines is due to numerical noise (statistics) and the 'incorrectness' of $W(M)$. $p_{\rm can}$ has been obtained from $p_{\rm muca}$ by reweighting.



- How accurately should W be determined? Remember that the simulation is formally correct with arbitrary $W(M)$. Good W can just save a lot of cpu-time. We should only demand that $p_{\mathrm{muca}}(M)$ does not become too small in any area of interest.

  $\mapsto$ If we allow p_{muca} to vary $\sim 50\%$, W can vary by $W \pm \log 2$.

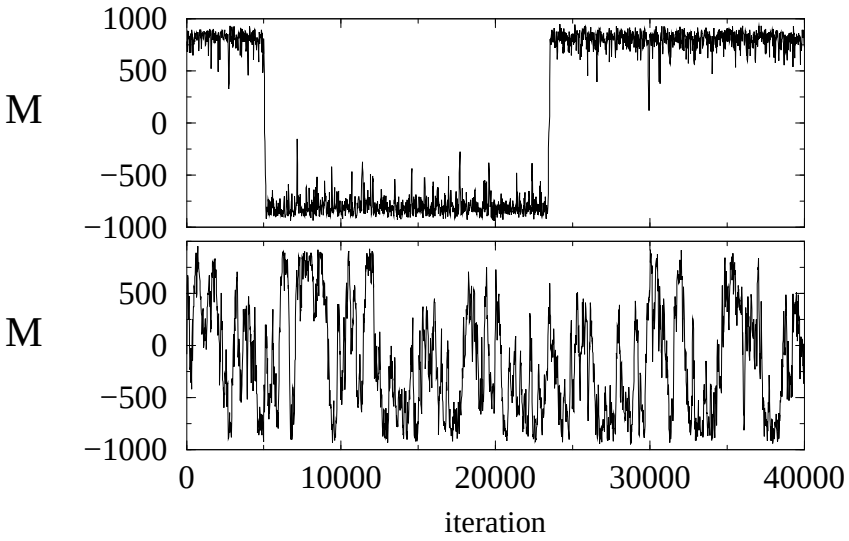
- The parametrization of $W(M)$ is not unique. Perhaps the most common is a piecewise linear one: choose a discrete small set $\{m_i\}$ of magnetizations, and parametrize $w_i = W(m_i)$, and interpolate linearly between m_i's:

$$W(M) = w_{i+1}\frac{M - m_i}{m_{i+1} - m_i} + w_i\frac{m_{i+1} - M}{m_{i+1} - m_i} \qquad \text{when } m_i \le M \le m_{i+1}\,.$$

- In the example above W was a function of M ("multimagnetic" ensemble). In general, W can be a function of any order parameter. Original formulation was in terms of E (useful for temperature-driven transitions).

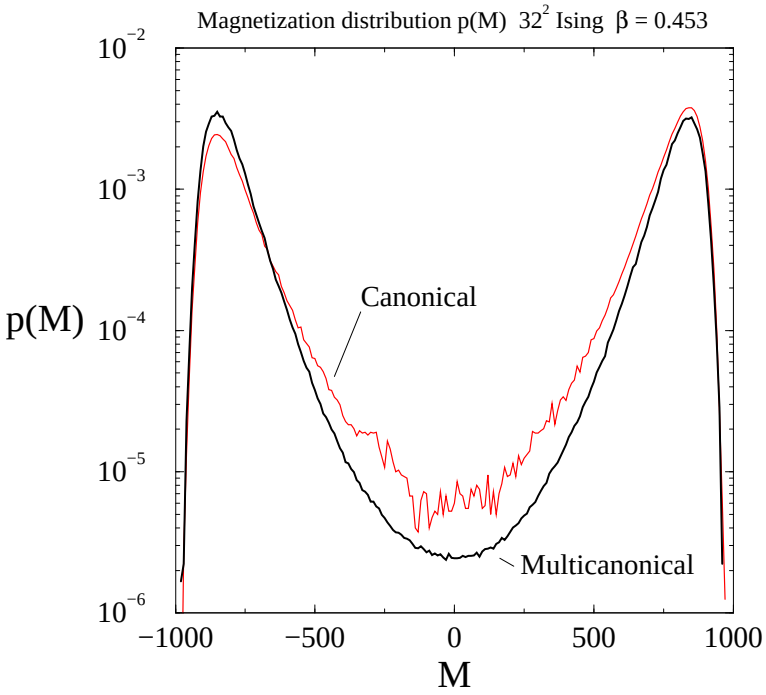
Performance comparison: again 32^2 Ising, at $\beta = 0.453$.

Monte Carlo time history of M, from canonical and multicanonical simulations:



The canonical simulation flips occasionally (rarely!) between the two phases, whereas the multicanonical does a random walk. The flip frequency is $\propto e^{-2\sigma L}$, where sigma is surface tension (later). Thus, it becomes exponentially more suppressed when L increases!

Canonical distributions $p_{\mathrm{can}}(M)$, obtained from multicanonical and canonical simulations (200000 measurements):



Another example:
20-state 2-d Potts model:

$E = -\sum_{ij} \delta_{s_i,s_j})$
$s_i = 1 \ldots 20$

In this case the transition is temperature-driven 1st order phase transition at
$\beta_c = \log(1+\sqrt{20}) \approx 2.900$
[A. Billoire, T. Neuhaus and B. Berg, 1994]

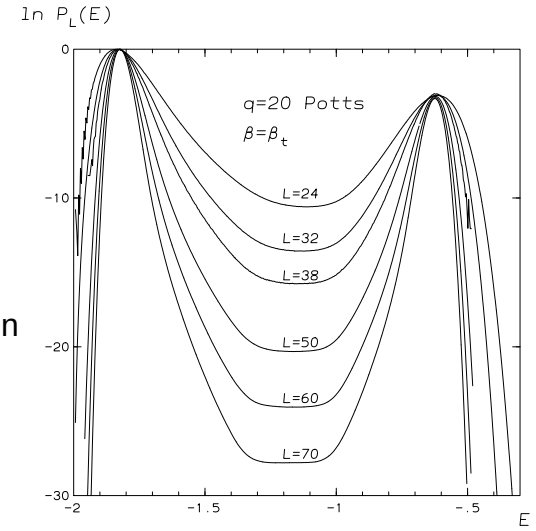


FIG. 8

Multicanonical simulations are not sensitive to the suppression seen in canonical simulations. However, they have to 'random walk' through the transition region. The region widens $\propto V$, but the 'step size' (change in one update cycle) increases only as $\propto V^{1/2}$. Thus, autocorrelation time τ behaves approximately as

- $\tau \sim e^{-\sigma 2L}$ in canonical simulations (supercritical slowing down)
- $\tau \sim V^2 = L^{2d}$ in multicanonical simulations

10.3 How to calculate W?

If we know $p_{\rm can}$, we know W – but $p_{\rm can}$ is the quantity we are trying to determine! To determine W:
1. Use canonical simulations to obtain rough $p_{\rm can}$
2. Finite-size scaling
3. Iterate: $W^1 \to W^2 \to \ldots$
4. Recursive computation of W: automatic iterative process. There are several alternatives for this; for example:

- Let us divide our order parameter M into bins, so that the system is in bin m when $M_m \le M < M_{m+1}$. The iteration is based on the frequency the system visits bin m during the Monte Carlo run.
- For presentational simplicity, I parametrize the weight function with a piecewise constant ansatz, $W(M) = w(m)$, when $M_m \le M < M_{m+1}$. Generalization to piecewise linear W is straightforward.

  The iteration proceeds as follows:

a) Set initial $w(m) = 0$.

b) During the Monte Carlo simulation, we subtract $w(m)$ every time we visit the bin m, thus reducing the probability of revisiting it. This can be achieved by modifying $w(m)$ every time we visit bin m by

$$w(m) \leftarrow w(m) - C$$

where initially constant $C = 1$ (or smaller). This forces the system quickly to visit all m-bins.

c) In order to make the iteration converge, after we have visited all m-bins we halve the value of C:

$$C \leftarrow C/2.$$

We continue with the new value of C until we again have visited all bins, then we halve C again and keep on repeating the cycle. Thus, the evolution of w becomes slower and slower, and it approaches a form which gives even distribution across the m-bins.

d) The procedure is halted when $w(m)$ has converged close enough to the right result, for example, when w changes at all m by less than some specified limit.

Note that $w(m)$ is uniformly decreasing in the above process; however, we can always add to $w(m)$ a constant independent of m (for example, we can renormalize first bin $w(0) = 0$).

The procedure above can be accelerated by starting from an initial guess for $w(m)$. However, in this case the initial value for C has to be small enough so that the initial guess is not immediately erased.

10.4 Simulated tempering

- Idea: upgrade β to dynamical variable; that is, allow it to change during a simulation.
- Canonical probability: $p_\beta(E) = n(E)e^{-\beta E + f_\beta}$
- Let now β_i be a set of β-values chosen so that $p_{\beta_i}(E) \equiv p_i(E)$ and $p_{i+1}(E)$ overlap.

1. Perform a (small) number of standard MC simulation steps at fixed $\beta = \beta_i$.
2. Propose a change $\beta_i \to \beta_j$. This is accepted with probability

$$p(\beta_i \to \beta_j) = \min\left(1\,,\,\frac{p_{\beta_j}(E)}{p_{\beta_i}(E)}\right) = \min\left(1\,,\ \exp[-(\beta_j - \beta_i)E + f_j - f_i]\right)$$

- Iterate steps 1 and 2.
- The update in 2. satisfies detailed balance. However, we do not a priori know the free energies f_i! These have to be estimated, as with the multicanonical algorithm.
- Note: f_i's are same parameters which appear in multiple histogram reweighting! So, we can perform simulations at fixed β_i's and use multiple histogram method to get f_i's.
- Advantage of simulated tempering over running with fixed temperatures: in simulated tempering, the system goes up and down in temperature - thus, it becomes periodically very disordered. This avoids it getting 'stuck' in one configuration at low temperatures.
- Same result can be achieved with multicanonical simulation with $W(E)$.
- Very suitable for spin glass simulations, polymers etc.

10.5 Parallel tempering

- Modification of simulated tempering idea:
- Again let β_i's, $i = 1 \ldots N$, be a set of β-values chosen so that $p_{\beta_i}(E) \equiv p_i(E)$ and $p_{i+1}(E)$ overlap.

1. Perform a (small) number of simulation steps simultaneously on N systems, with $\beta = \beta_1, \beta_2 \ldots \beta_N$ respectively. After the simulation, these systems will have energy values $E_1, E_2, \ldots$.
2. Propose a swap of beta-values $\beta_i \leftrightarrow \beta_j$. This is accepted with probability

$$\begin{aligned} p(\beta_i \leftrightarrow \beta_j) &= \min\left(1\,,\,\frac{P_{\text{after}}}{P_{\text{before}}}\right) = \min\left(1\,,\,\frac{e^{-\beta_i E_j} e^{-\beta_j E_i}}{e^{-\beta_i E_i} e^{-\beta_j E_j}}\right) \\ &= \min\left(1\,,\,\exp[-(\beta_i - \beta_j)(E_i - E_j)]\,\right) \end{aligned}$$

- Iterate steps 1 and 2.
- Does essentially the same trick as parallel tempering - applicable to similar problems.
- Advantages: no f_i's needed, thus, no tuning! Also, method is inherently parallel.

11 Cluster update algorithms

- Cluster update algorithms are the most succesful *global update* methods in use. These methods update the variables globally, in one step, whereas the standard local methods operate on one variable at a time.
- Another common method: **multigrid**. Used a lot in solving differential equations; however, multigrid has very limited success in Monte Carlo simulations.
- A global update can reduce the *autocorrelation time* of the update and thus greatly reduce the statistical errors.
- To repeat, we recall that the expectation values ($\beta \sim 1/T$):

$$\langle O \rangle_T = \frac{1}{Z} \int \prod_x d\phi_x \, O(\phi) \, e^{-\beta E(\phi)}$$

- With Monte Carlo simulation: generate a series of configurations $\Phi_1, \, \Phi_2 \, \ldots \, \Phi_N$ with some MC method. Measure $O_i = O(\Phi_i)$. Now

$$\frac{1}{N} \sum_i O_i \; \rightarrow \langle O \rangle, \quad \text{when } N \rightarrow \infty$$

- At finite N, the estimated error:

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

  if and only if the measurements O_i are statistically independent.
- However, in reality the update modifies the previous configuration (Markov chain). Thus, the successive configurations are correlated: $C(t) \propto \exp -t/\tau$. $\tau =$ auto-correlation time.
- # of independent configs is $\sim N/(2\tau) \rightarrow \delta O \approx \sqrt{2\tau} \delta O_{\rm naive}$.
- τ depends on the MC update algorithm: we want an update with as small τ as possible $\rightarrow$ Cluster algorithms.

11.1 Fundamentals: Fortuin-Kasteleyn cluster decomposition

Ising model (arbitrary dim.):

$$E = -\sum_{<i,j>} s_i s_j \qquad\qquad Z = \sum_{\{s\}} e^{-\beta E} \, .$$

($s_i = \pm 1$, $< i, j >$ nearest neighbour sites).

Note: previously we defined E with a factor $\frac{1}{2}$ in front. Thus, $\beta_{\rm here} = \beta_{\rm prev.}/2$, and in 2 dimensions $\beta_c = 1/2 \log(1 + \sqrt{2}) \approx 0.44$. This way is more conventional; the transformation between normalizations is trivial.

Consider interaction between fixed n.n.-sites $< l, m >$, and remove it from E:

$$E_{l,m} = -\sum_{<i,j> \neq <l,m>} s_i s_j \, .$$

Define now partition functions where s_i, s_j are equal or different:

$$Z^{\rm same}_{l,m} \equiv \sum_{\{s\}} \delta_{s_l,s_m} e^{-\beta E_{l,m}} \, , \qquad Z^{\rm diff.}_{l,m} \equiv \sum_{\{s\}} (1 - \delta_{s_l,s_m}) e^{-\beta E_{l,m}} \, .$$

Now we can clearly write the original partition function as

$$Z = e^{\beta} Z^{\rm same}_{l,m} + e^{-\beta} Z^{\rm diff.}_{l,m}$$

Furthermore, defining

$$Z^{\rm ind.}_{l,m} \equiv \sum e^{-\beta E_{l,m}} = Z^{\rm same}_{l,m} + Z^{\rm diff.}_{l,m} \, .$$

the partition function finally becomes

$$Z = (e^{\beta} - e^{-\beta}) Z^{\rm same}_{l,m} + e^{-\beta} Z^{\rm ind.}_{l,m}$$

Remember that the partition function is $Z = \sum_{\{s\}} p(\{s\})$. Thus, since $Z^{\rm same}$ contains only configs where $s_l = s_m$, and $Z^{\rm ind.}$ contains no restriction or input for s_i, s_j-link, the weighting factors of Z's can be considered as relative probabilities of a **bond** between sites l, m and no bond (= independent states).

Normalizing the probabilities:

$$\begin{aligned} p_{\rm bond} &= 1 - e^{-2\beta} \\ p_{\rm ind.} &= e^{-2\beta} = 1 - p_{\rm bond} \end{aligned}$$

Repeating for all $< i, j >$,
a) sites linked to each other by bonds form clusters
b) different clusters are independent: $s_c = \pm 1$

and the partition function can be written as

$$Z = \sum_{\substack{\text{all bond} \\ \text{configurations}}} \sum_{s_c = \pm 1} p^b (1-p)^n = \sum_{\substack{\text{all bond} \\ \text{configurations}}} p^b (1-p)^n 2^{N_c}$$

Here

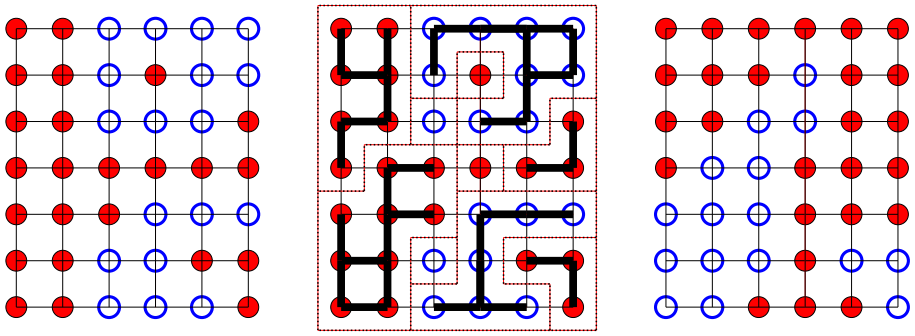
$$\begin{aligned} p &= p_{\text{bond}} = 1 - p_{\text{ind.}} = 1 - e^{-2\beta} \\ b &= \textbf{\# of interactions that form a bond}, \\ n &= \textbf{\# of interactions that do not form a bond}, \\ N_c &= \text{\# of clusters.} \end{aligned}$$

This decomposition is at the core of the cluster algorithms.

11.2 Swendsen-Wang cluster update

[Swendsen and Wang, PRL 58 (1987) 86]
Beginning with an arbitary configuration s_i, one SW cluster update cycle is:

1. Inspect all nn-states s_i, s_j. If $s_i = s_j$, create a bond between sites i, j with probability $p = 1 - \exp(-2\beta)$ (otherwise no bond).
2. Construct clusters $=$ sets of points connected by bonds.
3. Set each cluster to a *random* value ± 1.



The configuration changes a lot in one update!
Is this a valid update? It satisfies

a) ergodicity (obvious)

b) detailed balance: $\frac{P(A \mapsto B)}{P(B \mapsto A)} = \exp -\beta(E_B - E_A)$?

Proof: consider $A \mapsto \mathcal{C} \mapsto B$, where $\mathcal{C}$ is some bond configuration compatible with both A and B. Since the clusters in $\mathcal{C}$ are independent, $P(\mathcal{C} \mapsto A) = P(\mathcal{C} \mapsto B) = 1/2^{N_c}$.

Now,

$$\frac{P(A \mapsto \mathcal{C})}{P(B \mapsto \mathcal{C})} = \frac{p^b\,(1-p)^{d_A}}{p^b\,(1-p)^{d_B}} = \exp[-\beta(E_B - E_A)]$$

where $d_{A,B}$ are the numbers of similar nn-states which are not connected by a bond. The last step comes from $E_A = \text{dim} \times V - 2(b + d_A)$. Thus $A \mapsto \mathcal{C} \mapsto B$ and $B \mapsto \mathcal{C} \mapsto A$ satisfy detailed balance for arbitrary $\mathcal{C}$, and the total transition probabilities $A \mapsto B$, $B \mapsto A$ must do it also.

11.3 Wolff single cluster update

[U. Wolff, PRL 62 (1989) 361]

Principle: do the cluster decomposition as in S-W, but invert ('flip') only one randomly chosen cluster! In practice:

1. Choose random site i.
2. Study neighbouring sites j. If $s_j = s_i$, join site j to cluster with probability $p = 1 - \exp(-2\beta)$.
3. Repeat step 2 for site j, if it was joined to the cluster. Keep on doing this as long as the cluster grows.
4. When the cluster is finished, invert the spins which belong to it.

• Usually slightly more effective than S-W (the average size of the clusters is larger. Why?).
• The minimum cluster size = 1, maximum = volume.
• Nicely recursive.
• Satisfies detailed balance.

11.4 Cluster update in c-pseudocode:

On the next page is a c-code snippet which performs Wolff cluster update.

• `s[loc]`: spin array, and `dran()` is a random number generator which returns a number fro 0 to 1.

• `n_neighbours` = $2 \times d$ is the number of neighbours a site has.

• `neighbour[dir][loc]` is the neighbour array: it gives the index of the site `loc` to direction `i`.

• `s[loc]` is flipped during the cluster growth: this prevents revisiting sites already included in the cluster.

• Recursion (as shown) is neat but not efficient! Better to reimplement the recursion with loops and temporary arrays.

```
void update_cluster()
{
  int start,state;

  start = dran() * volume;    /* starting location */
  state = s[start];           /* starting spin value */
  /* start growing and inverting the cluster */
  grow_cluster(start, state);
}

void grow_cluster(int loc,int state)
{
  int dir,new_loc;

  s[loc] = -s[loc];        /* invert the spin at this location */

  /* begin loop over neighbour locations */
  for (dir=0; dir<n_neighbours; dir++) {
    new_loc = neighbour[dir][loc];   /* neighbour index */

    if (s[new_loc] == state && exp(-2.0*beta) < dran())
      grow_cluster(new_loc,state);
  }
}
```

- The presented algorithm for growing the cluster is a **depth-first** algorithm: it starts from the root and follows the cluster tree branch as far as possible, before taking another branch.
- Another option is **width-first**:

  1. Start from the root.
  2. Mark all neighbouring points which are accepted to the cluster.
  3. Looping through all these points, mark all points where the cluster grows further.
  4. Continue from 3. for these new points, until no more points are accepted.

- Both the depth-first and width-first require comparable amount of information, and are about as good. Width-first is somewhat easier to program, though.
- However, for Swendsen-Wang update all of the clusters have to be found. In this case there exists an optimized cluster search algorithm by Hoshen and Kopelmann.

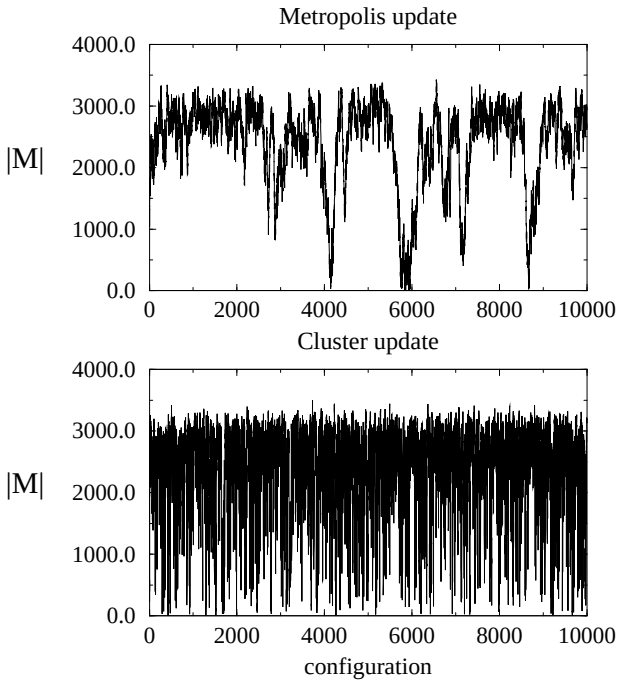
As an example, let us consider 2-dimensional Ising model. It has 2nd order phase transition at $\beta = 1/2 \log(1 + \sqrt{2}) \approx 0.44$ (Curie point).
Correlation length (and cluster size!) diverges when one approaches the critical point.

Compare the performance of
a) Metropolis (typewriter ordering) and
b) Wolff cluster algorithm.

System size 64^2, $\beta = \beta_c \approx 0.44$. Compare measurements of absolute value of magnetization

$$|M| = |\sum_i s_i|$$

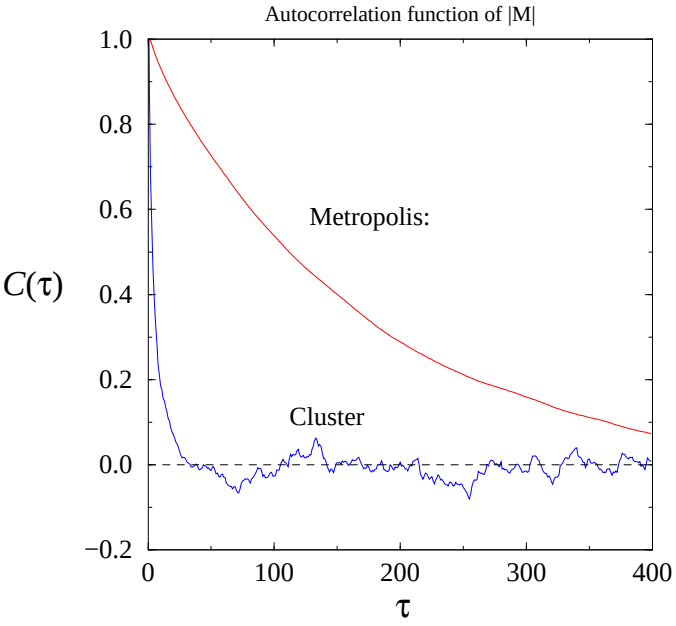
Cluster update covers the phase space much faster than Metropolis! (which was the best of the "single-site" update methods).

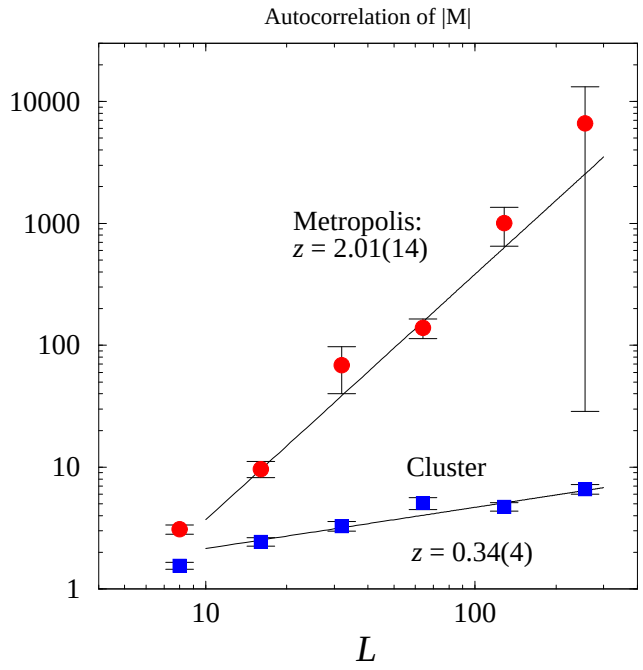


The autocorrelation function describes how fast measurements become decorrelated.

Let now $O_i = |M|_i - < |M| >$, where $|M|_i$ is the measurement #i of $|M|$. Autocorrelation function $C(t)$ of $|M|$:

$$C(t) \equiv \frac{\frac{1}{N-t}\sum_i O_i O_{i+t}}{\langle O^2 \rangle} \propto e^{-t/\tau}$$





• *Exponential autocorrelation time:* the exponential decay length of $C(t)$ at large t.

• *Integrated autocorrelation time:* $\tau_{\text{int.}} \equiv \frac{1}{2} + \sum_{t=1}^{\infty} C(t)$.

Shown here is the integrated autocorrelation time $\tau_{\text{int.}}$ measured from different lattice sizes $L^2 = 8^2$ – 256^2 at β_c.

• Expected behaviour: $\tau_a \propto L^z$, where z is the dynamical critical exponent. For local (dissipative) algorithms, $z \gtrsim 2$.

• For volumes $\sim 256^2$, cluster algorithm is ~ 1000 times better than Metropolis!

• Measured in real (cpu/wallclock) time, cluster is even better (at the critical temperature $\beta_c = 0.44$):
time/update $\propto V$ (local Metropolis)
time/update $\propto V^x$, where $x < 1$

Using the example of section 3.14, 2d Ising, volume 64^2, and measuring the integrated autocorrelation time of E:

	$\tau_{\rm int.}$	time(ms)/iteration	time(ms)/iter. $\times 2\tau_{\rm int}$
Cluster	5.7	0.28	3.2
Metro, typew.	54	1.0	108
HB, typew.	157	1.2	364
Metro, random	271	1.4	748
HB, random	316	1.6	1004

11.5 What about other models?

Cluster algorithms can work, if one can embed an Ising system into the original system.
Example: O(N) non-linear sigma model:
The Hamiltonian of the model is

$$E = -\sum_{<ij>} \sigma_i \cdot \sigma_j \,, \qquad \text{where} \quad |\sigma_i|^2 = \sum_{\alpha=1}^{N} (\sigma_i^\alpha)^2 = 1$$

Thus, spins σ_i are N-dimensional vectors of unit length.
Embedding: [U.Wolff, PRL 62 (1989) 361]
Map arbitrary sigma model configuration to Ising model with Hamiltonian

$$E_I = -\sum_{<ij>} J_{ij} s_i s_j$$

as follows:

- Choose a random O(N) vector r.
- Set all Ising model links $J_{ij} = (r \cdot \sigma_i)(r \cdot \sigma_j)$.
- Ising update $s_i \to -s_i$ corresponds to reflection of σ_i along the vector r: $\sigma_i \to \sigma_i - 2(r \cdot \sigma_i)\, r$
- Initially, all $s_i = +1$.

After the mapping, we can use either S-W or Wolff update on the Ising spins. In practice, one Wolff update cycle proceeds as follows:

1. Choose random O(N) vector r.
2. Choose random site i as the starting point for the cluster.
3. Study neighbouring sites j. Join site j to cluster with probability $p = 1 - \exp(-2\beta J_{ij})$, where $J_{ij} = (r \cdot \sigma_i)(r \cdot \sigma_j)$.
4. Repeat step 3 for all sites joined to the cluster. Keep on doing this as long as the cluster grows.

5. When the cluster is finished, reflect $\sigma_i \to \sigma_i - 2(r \cdot \sigma_i)$ for all sites which belong to the cluster.

Typically the reflection is performed during the cluster growth.

Single cluster O(N) sigma model algorithm works even better than for Ising: In 3-dimensional O(4) sigma model, the dynamical critical exponent z becomes negative.

11.5.1 In what models clusters work, in what models they fail?

Clusters (usually) fail, if
• there are frustrated couplings (spin glasses, gauge theories . . .)
• one cannot construct a symmetric reflection operation
• spins are 'frozen' in place by external fields etc.

Cluster updates are (normally) usable only in proximity of a 2nd order phase transitions: large correlation lengths $\to$ large clusters.

Nevertheless, sometimes they are useful when correlation lengths are finite but still large ($\gg 1$) in lattice units.

More conditions:

In order to work, the reflection operation $R\sigma = \sigma'$ must satisfy the following: the set of the fixed points of the reflection ($R\sigma = \sigma$) must separate the phase space of σ in two disconnected sets.
For example, in O(3), the fixed points are the points with $r \cdot \sigma = 0$, i.e. the points on the 'equator' perpendicular to the reflection vector r. This divides O(3) into disconnected 'northern' and 'southern' hemisphere.
Mathematically: fixed point set should have codimension 1 [Lüscher, Weisz, Wolff, NPB 359 (1991) 221].

11.6 Reduced variance

- Clusters can also substantially reduce statistical noise in some measurements – this is in addition to the acceleration in update speed (*reduced variance*).

- For example, consider spin-spin correlation function

$$G(i,j) = s_i s_j \qquad\qquad \langle G(i,j) \rangle \sim e^{-|i-j|/\xi}$$

- Using normal MC averaging over spin configurations (whichever algorithm we use), the variance of G is

  $$\langle G^2 \rangle - \langle G \rangle^2 \approx 1 - e^{2|i-j|/\xi} \approx 1$$

  Thus, the absolute error is $\sim$ constant independent of $|i-j|$, and signal/error$\sim e^{-|i-j|/\xi}$.

- However, consider the expectation value using the cluster partition function

  $$\langle G(i,j) \rangle = \frac{1}{Z} \sum_{\text{bonds}} p^b (1-p)^n \sum_{s_{\text{clust.}}} s_i s_j$$

  The last sum is clearly

  $$\sum_{s_{\text{clust.}}} s_i s_j = \begin{cases} 1 & \text{if } s_i \text{ and } s_j \text{ belong to the same cluster} \\ 0 & \text{otherwise.} \end{cases}$$

- Thus, in a (Swendsen-Wang) cluster MC update/measurement, we measure

  $$\langle s_i s_j \rangle_{SW} = \left\langle \sum_{\text{clust.}} \Theta_c(i) \Theta_c(j) \right\rangle_{SW}$$

  where $\Theta_c(i) = 1$ if point i belongs to cluster c, 0 otherwise. Since the fraction of 1 to 0's must be $e^{-|i-j|/\xi}$, the variance is

  $$(\langle G^2 \rangle - \langle G \rangle^2)_{SW} \approx e^{-|i-j|/\xi} - e^{-2|i-j|/\xi} \approx e^{-|i-j|/\xi}$$

  The absolute error $\sim e^{-|i-j|/2\xi}$, and signal/error$\sim e^{-|i-j|/2\xi}$.

- An exponential factor is gained over the 'naive method! This is due to the fact that the cluster measurement effectively sums over all possible spin configurations compatible with the cluster decomposition.

- Another example: magnetic susceptibility

  $$\chi_M = \langle M^2 \rangle = \frac{1}{V} \sum_{i,j} s_i s_j = \frac{1}{V} \sum_{\text{clust.}} N_c^2$$

  where N_c is the cluster size, and we assume $\langle M \rangle = 0$ ($T > T_c$).

- This readily generalizes to other spin observables. However, 3- or higher point functions do not usually gain from cluster measurements, nor observables which depend on energy. The operators rapidly become quite complicated, for example,

$$\langle s_i s_j s_k s_l \rangle = \theta(i,j,k,l) + \ \theta(i,j)\theta(k,l) + \theta(i,k)\theta(j,l) + \theta(i,l)\theta(j,k).$$

- The above was for Swendsen-Wang type update. What about the Wolff single cluster update? In this case, the clusters are chosen with biased probability $p_c = N_c/V$. This we can compensate by multiplying the observable with $1/p_c$, and thus

$$\langle s_i s_j \rangle_{1C} = \left\langle \frac{V}{N_c} \Theta_c(i) \Theta_c(j) \right\rangle_{1C}$$

- Likewise, the single-cluster susceptibility measurement becomes

$$\chi_M = \langle M^2 \rangle = \langle N_c \rangle_{1C}$$

- Through the Ising embedding, the reduced variance measurements can be done also for other models.

12 Final notes

Monte Carlo simulation is a versatile method for studying thermodynamical properties in a very wide variety of frameworks. It is also widely used in studying (quantum) field theories.

In this course we have studied the main framework of Monte Carlo simulation methods. The central parts of the course are:

– Sect. 2: Monte Carlo integration, importance sampling, convergence

– Sect. 3: generation of random numbers from given distributions (not so much the basic generators, albeit it is good to know the weaknesses)

– Sect. 4 (& 5): detailed balance, Metropolis/heat bath etc. updates, applied to various models, autocorrelation, autocorrelation time, error estimation

Thus, given almost any (potential) energy functional (must be bounded from below!), the methods above should be sufficient to construct a valid update algorithm and measure quantities of interest!

In further sections we discussed more specialized analysis- and simulation methods. Of these, reweighting (Sect. 6) and bootstrap (or jackknife) (Sect. 7) are in very common use.

What we did not discuss?

In this course we kept to simple “spin models” and potential systems. In general, the interaction energy can be much more complicated (e.g. quantum mechanical fermion systems), requiring quite different Monte Carlo methods.

There are also several methods which apply Monte Carlo -like methodology but the system does not remain in equilibrium. These include *simulated annealing*, where the temperature is varied up and down during the simulation.

In this course we discussed only static problems, i.e. not dependent on time. Time evolution opens up a vast variety of methods, which can be more or less related to Monte Carlo.
For example, in some cases it is possible to associate the Monte Carlo simulation time with the real time. These processes are fully dissipative. In another end of the spectrum the time evolution is fully deterministic (solving equations of motion; molecular dynamics). There also exist methods which have some degree of noisiness included in the equations of motion (Langevin-like methods).