Classical and Quantum Monte Carlo Algorithms and Exact Diagonalization

Matthias Troyer, ETH Zürich
`troyer@phys.ethz.ch`

July 29, 2004

Contents

1 Introduction

These lecture notes provide an introduction to classical and quantum Monte Carlo simulations for magnetic models, with an emphasis on non-local updates, as well as an introduction to exact diagonalization. These notes are based in part on a course on algorithms for many-body problems taught at ETH Zürich from 1999-2004. For details of these methods and for references to the original literature I refer to the references in the review that will also be handed out.

2 Monte Carlo integration

In thermodynamics, as in many other fields of physics, often very high dimensional integrals have to be evaluated. Even in a classical N-body simulation the phase space has dimension $6N$, as there are three coordinates each for the location and position of each particle. In a quantum mechanical problem of N particles the phase space is even exponentially large as a function of N.

2.1 Standard integration methods

A Riemannian integral $f(x)$ over an interval $[a, b]$ can be evaluated by replacing it by a finite sum:

$$\int_a^b f(x)dx = \sum_{i=1}^{N} f(a + i\Delta x)\Delta x + \mathrm{O}(\Delta x^2), \tag{1}$$

where $\Delta x = (a - b)/N$. The discretization error decreases as $1/N$ for this simple formula. Better approximations are the trapezoidal rule

$$\int_a^b f(x)dx = \Delta x \left[\frac{1}{2}f(a) + \sum_{i=1}^{N-1} f(a + i\Delta x) + \frac{1}{2}f(b)\right] + \mathrm{O}(\Delta x^2), \tag{2}$$

or the Simpson rule

$$\begin{aligned}\int_a^b f(x)dx &= \frac{\Delta x}{3}\left[f(a) + \sum_{i=1}^{N/2} 4f(a + (2i-1)\Delta x)\right. \\ &\left. + \sum_{i=1}^{N/2-1} 2f(a + 2i\Delta x) + f(b)\right] + \mathrm{O}(\Delta x^4),\end{aligned} \tag{3}$$

which scales like N^{-4}.

For more elaborate schemes like the Romberg method or Gaussian integration we refer to textbooks.

In higher dimensions the convergence is much slower though. With N points in d dimensions the linear distance between two points scales only as $N^{-1/d}$. Thus the Simpson rule in d dimensions converges only as $N^{-4/d}$, which is very slow for large d. The solution are Monte Carlo integrators.

2.2 Monte Carlo integrators

With randomly chosen points the convergence does not depend on dimensionality. Using N randomly chosen points $\mathbf{x}_i$ the integral can be approximated by

$$\frac{1}{\Omega}\int f(\mathbf{x})d\mathbf{x} \approx \overline{f} := \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i), \tag{4}$$

where $\Omega := \int d\mathbf{x}$ is the integration volume. As we saw in the previous chapter the errors of such a Monte Carlo estimate the errors scale as $N^{-1/2}$. In $d \geq 9$ dimensions Monte Carlo methods are thus preferable to a Simpson rule.

2.2.1 Importance Sampling

This simple Monte Carlo integration is however not the ideal method. The reason is the variance of the function

$$\mathrm{Var} f = \Omega^{-1}\int f(\mathbf{x})^2 d\mathbf{x} - \left[\Omega^{-1}\int f(\mathbf{x})d\mathbf{x}\right]^2 \approx \frac{N}{N-1}(\overline{f^2} - \overline{f}^2). \tag{5}$$

The error of the Monte Carlo simulation is

$$\Delta = \sqrt{\frac{\mathrm{Var} f}{N}} \approx \sqrt{\frac{\overline{f^2} - \overline{f}^2}{N-1}}. \tag{6}$$

In phase space integrals the function is often strongly peaked in a small region of phase space and has a large variance. The solution to this problem is “importance sampling”, where the points $\mathbf{x}_i$ are chosen not uniformly but according to a probability distribution $p(\mathbf{x})$ with

$$\int p(\mathbf{x})d\mathbf{x} = 1. \tag{7}$$

Using these p-distributed random points the sampling is done according to

$$\langle f \rangle = \Omega^{-1}\int A(\mathbf{x})d\mathbf{x} = \Omega^{-1}\int \frac{f(\mathbf{x})}{p(\mathbf{x})}p(\mathbf{x})d\mathbf{x} \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)} \tag{8}$$

and the error is

$$\Delta = \sqrt{\frac{\mathrm{Var} f/p}{N}}. \tag{9}$$

It is ideal to choose the distribution function p as similar to f as possible. Then the ratio f/p is nearly constant and the variance small.

As an example, the function $f(x) = \exp(-x^2)$ is much better integrated using exponentially distributed random numbers with $p(x) = \exp(-\lambda x)$ instead of uniformly distributed random numbers.

A natural choice for the weighting function p is often given in the case of phase space integrals or sums, where an observable A is averaged over all configurations $\mathbf{x}$ in phase space where the probability of a configuration is $p(\mathbf{x})$. The phase space average $\langle A \rangle$ is:

$$\langle A \rangle = \frac{\int A(\mathbf{x}) p(x) d\mathbf{x}}{\int p(x) d\mathbf{x}}. \tag{10}$$

2.3 Markov chains and the Metropolis algorithm

In general problems with arbitrary distributions p it will not be possible to create p-distributed configuration from scratch. Instead a Markov process can be used.

Starting from an initial point $\mathbf{x}_0$ a Markov chain of states is generated:

$$\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2 \rightarrow \ldots \rightarrow \mathbf{x}_n \rightarrow \mathbf{x}_{n+1} \rightarrow \ldots \tag{11}$$

A transition matrix $W_{\mathbf{xy}}$ gives the transition probabilities of going from state $\mathbf{x}$ to state $\mathbf{y}$ in one step of the Markov process. As the sum of probabilities of going from state $\mathbf{x}$ to any other state is one, the columns of the matrix W are normalized:

$$\sum_{\mathbf{y}} W_{\mathbf{xy}} = 1 \tag{12}$$

A consequence is that the Markov process conserves the total probability. Another consequence is that the largest eigenvalue of the transition matrix W is 1 and the corresponding eigenvector with only positive entries is the equilibrium distribution which is reached after a large number of Markov steps.

We want to determine the transition matrix W so that we asymptotically reach the desired probability $p_{\mathbf{x}}$ for a configuration i. A set of sufficient conditions is:

1. **Ergodicity:** It has to be possible to reach any configuration $\mathbf{x}$ from any other configuration $\mathbf{y}$ in a finite number of Markov steps. This

means that for all $\mathbf{x}$ and $\mathbf{y}$ there exists a positive integer $n < \infty$ such that $(W^n)_{\mathbf{xy}} \neq 0$.

2. **Detailed balance:** The probability distribution $p_{\mathbf{x}}^{(n)}$ changes at each step of the Markov process:

$$\sum_{\mathbf{x}} p_{\mathbf{x}}^{(n)} W_{\mathbf{xy}} = p_{\mathbf{y}}^{(n+1)}. \tag{13}$$

but converges to the equilibrium distribution $p_{\mathbf{x}}$. This equilibrium distribution $p_{\mathbf{x}}$ is an eigenvector with left eigenvalue 1 and the equilibrium condition

$$\sum_{\mathbf{x}} p_{\mathbf{x}} W_{\mathbf{xy}} = p_{\mathbf{y}} \tag{14}$$

must be fulfilled. It is easy to see that the detailed balance condition

$$\frac{W_{\mathbf{xy}}}{W_{\mathbf{yx}}} = \frac{p_{\mathbf{y}}}{p_{\mathbf{x}}} \tag{15}$$

is sufficient.

The simplest Monte Carlo algorithm is the Metropolis algorithm:

- Starting with a point $\mathbf{x}_i$ choose randomly one of a fixed number N of changes $\Delta\mathbf{x}$, and propose a new point $\mathbf{x}' = \mathbf{x}_i + \Delta\mathbf{x}$.

- Calculate the ratio os the probabilities $P = p_{\mathbf{x}'}/p_{\mathbf{x}_i}$.

- If $P > 1$ the next point is $\mathbf{x}_{i+1} = \mathbf{x}'$

- If $P < 1$ then $\mathbf{x}_{i+1} = \mathbf{x}'$ with probability P, otherwise $\mathbf{x}_{i+1} = \mathbf{x}_i$. We do that by drawing a random number r uniformly distributed in the interval $[0, 1[$ and set $\mathbf{x}_{i+1} = \mathbf{x}'$ if $r < P$.

- Measure the quantity A at the new point $\mathbf{x}_{i+1}$.

The algorithm is ergodic if one ensures that the N possible random changes allow all points in the integration domain to be reached in a finite number of steps. If additionally for each change $\Delta\mathbf{x}$ there is also an inverse change $-\Delta\mathbf{x}$ we also fulfill detailed balance:

$$\frac{W_{ij}}{W_{ji}} = \frac{\frac{1}{N}\min(1, p(j)/p(i))}{\frac{1}{N}\min(1, p(i)/p(j))} = \frac{p(j)}{p(i)}. \tag{16}$$

As an example let us consider summation over integers i. We choose $N = 2$ possible changes $\Delta i{=}\pm 1$ and fulfill both ergodicity and detailed balance as long as $p(i)$ is nonzero only over a finite contiguous subset of the integers.

To integrate a one-dimensional function we take the limit $N \to \infty$ and pick any change $\delta \in [-\Delta, \Delta]$ with equal probability. The detailed balance equation (16) is only modified in minor ways:

$$\frac{W_{ij}}{W_{ji}} = \frac{\frac{d\delta}{2\Delta}\min(1, p(j)/p(i))}{\frac{d\delta}{2\Delta}\min(1, p(i)/p(j))} = \frac{p(j)}{p(i)}. \tag{17}$$

Again, as long as $p(x)$ is nonzero only on a finite interval detailed balance and ergodicity are fulfilled.

2.4 Autocorrelations, equilibration and Monte Carlo error estimates

2.4.1 Autocorrelation effects

In the determination of statistical errors of the Monte Carlo estimates we have to take into account correlations between successive points $\mathbf{x}_i$ in the Markov chain. These correlations between configurations manifest themselves in correlations between the measurements of a quantity A measured in the Monte Carlo process. Denote by $A(t)$ the measurement of the observable A evaluated at the t-th Monte Carlo point $\mathbf{x}_t$. The autocorrelations decay exponentially for large time differences Δ:

$$\langle A_t A_{t+\Delta}\rangle - \langle A\rangle^2 \propto \exp(-\Delta/\tau_A^{(exp)}) \tag{18}$$

Note that the autocorrelation time τ_A depends on the quantity A.

An alternative definition is the integrated autocorrelation time $\tau_A^{(int)}$, defined by

$$\tau_A^{(int)} = \frac{\sum_{\Delta=1}^{\infty}\left(\langle A_t A_{t+\Delta}\rangle - \langle A\rangle^2\right)}{\langle A^2\rangle - \langle A\rangle^2} \tag{19}$$

As usual the expectation value of the quantity A can be estimated by the mean:

$$\overline{A} \equiv \frac{1}{N}\sum_i = 1^N A_i \tag{20}$$

The error estimate

$$\Delta A = \sqrt{\frac{\mathrm{VarA}}{N}} \tag{21}$$

has to be modified because consecutive measurements are correlated . The error estimate $(\Delta A)^2$ is calculates as the expectation value of the squared difference between sample average and expectation value:

$$\begin{aligned}
(\Delta A)^2 &= \langle(\overline{A}-\langle A\rangle)^2\rangle = \langle\left(\frac{1}{N}\sum_{t=1}^{N} A(t) - \langle A\rangle\right)^2\rangle \\
&= \frac{1}{N^2}\sum_{i=1}^{N}\left(\langle A(t)^2 - \langle A\rangle^2\ \rangle\right) \\
&\quad + \frac{2}{N^2}\sum_{t=1}^{N}\sum_{\Delta=1}^{N-t}\left(\langle A(t)A(t+\Delta)\rangle - \langle < A >^2\rangle\right) \\
&\approx \frac{1}{N}\mathrm{Var}A\ (1+2\tau_A^{(int)}) \\
&\approx \frac{1}{N-1}\langle\overline{A^2}-\overline{A}^2\rangle(1+2\tau_A^{(int)}) \qquad (22)
\end{aligned}$$

In going from the second to third line we assumed $\tau_A^{(int)} \ll N$ and extended the summation over Δ to infinity. In the last line we replaced the variance by an estimate obtained from the sample. We see that the number of statistical uncorrelated samples is reduced from N to $N/(1+2\tau_A^{(int)})$.

In many Monte Carlo simulations the error analysis is unfortunately not done accurately. Thus we wish to discuss this topic here in more detail.

2.4.2 The binning analysis

The binning analysis is a reliable way to estimate the integrated autocorrelation times. Starting from the original series of measurements $A_i^{(0)}$ with $i = 1, \ldots, N$ we iteratively create "binned" series by averaging over to consecutive entries:

$$A_i^{(l)} := \frac{1}{2}\left(A_{2i-1}^{(l-1)} + A_{2i}^{(l-1)}\right), \qquad i = 1, \ldots, N_l \equiv N/2^l. \qquad (23)$$

These bin averages $A_i^{(l)}$ are less correlated than the original values $A_i^{(0)}$. The mean value is still the same.

The errors $\Delta A^{(l)}$, estimated incorrectly using equation (21)

$$\Delta A^{(l)} = \sqrt{\frac{\mathrm{Var}A^{(l)}}{N_l - 1}} \approx \frac{1}{N_l}\sqrt{\sum_{i=1}^{N_l}\left(A_i^{(l)} - \overline{A^{(l)}}\right)^2} \qquad (24)$$

however increase as a function of bin size 2^l. For $2^l \gg \tau_A^{(int)}$ the bins become uncorrelated and the errors converge to the correct error estimate:

$$\Delta A = \lim_{l\to\infty} \Delta A^{(l)}. \qquad (25)$$

This binning analysis gives a reliable recipe for estimating errors and autocorrelation times. One has to calculate the error estimates for different bin sizes l and check if they converge to a limiting value. If convergence is observed the limit ΔA is a reliable error estimate, and $\tau_A^{(int)}$ can be obtained from equation (22) as

$$\tau_A^{(int)} = \frac{1}{2}\left[\left(\frac{\Delta A}{\Delta A^{(0)}}\right)^2 - 1\right] \tag{26}$$

If however no convergence of the $\Delta A^{(l)}$ is observed we know that $\tau_A^{(int)}$ is longer than the simulation time and we have to perform *much* longer simulations to obtain reliable error estimates.

To be really sure about convergence and autocorrelations it is very important to start simulations always on tiny systems and check convergence carefully before simulating larger systems.

This binning analysis is implemented in the ALPS library which will be used in the hands-on session.

2.4.3 Jackknife analysis

The binning procedure is a straightforward way to determine errors and autocorrelation times for Monte Carlo measurements. For functions of measurements like $U = \langle A \rangle / \langle B \rangle$ it becomes difficult because of error propagation and cross-correlations.

Then the jackknife procedure can be used. We again split the measurements into M bins of size $N/M \gg \tau^{(int)}$ that should be much larger than any of the autocorrelation times.

We could now evaluate the complex quantity U in each of the M bins and obtain an error estimate from the variance of these estimates. As each of the bins contains only a rather small number of measurements N/M the statistics will not be good. The jackknife procedure instead works with $M+1$ evaluations of U. U_0 is the estimate using all bins, and U_i for $i = 1, \ldots M$ is the value when all bins *except* the i-th bin are used. That way we always work with a large data set and obtain good statistics.

The resulting estimate for U will be:

$$U = U_0 - (M-1)(\overline{U} - U_0) \tag{27}$$

with a statistical error

$$\Delta U = \sqrt{M-1}\left(\frac{1}{M}\sum_{i=1}^{M}(U_i)^2 - (\overline{U})^2\right)^{1/2}, \tag{28}$$

where

$$\overline{U} = \frac{1}{M}\sum_{i=1}^{M} U_i, \tag{29}$$

The jackknife analysis is implemented in the ALPS library which will be used in the hands-on session.

2.4.4 Equilibration

Thermalization is as important as autocorrelations. The Markov chain converges only asymptotically to the desired distribution. Consequently, Monte Carlo measurements should be started only after a large number N_{eq} of equilibration steps, when the distribution is sufficiently close to the asymptotic distribution. N_{eq} has to be much larger than the thermalization time which is defined similar to the autocorrelation time as:

$$\tau_A^{(eq)} = \frac{\sum_{\Delta=1}^{\infty}(\langle A_0 A_\Delta\rangle - \langle A\rangle^2)}{\langle A_0\rangle\langle A\rangle - \langle A\rangle^2} \tag{30}$$

It can be shown that the thermalization time is the maximum of all autocorrelation times for all observables and is related to the second largest eigenvalue Λ_2 of the Markov transition matrix W by $\tau^{(th)} = -1/\ln\Lambda_2$. It is recommended to thermalize the system for at least ten times the thermalization time before starting measurements.

3 Classical Monte Carlo simulations

Before getting to algorithms for quantum systems we first review the corresponding algorithms for classical systems, starting with the Ising model as the simplest model.

3.1 The Ising model

The Ising model is the simplest model for a magnetic system and a prototype statistical system. We will use it for our discussion of thermodynamic phase transitions. It consists of an array of classical spins $\sigma_i = \pm 1$ that can point either up ($\sigma_i = +1$) or down ($\sigma_i = -1$). The Hamiltonian is

$$H = -J\sum_{\langle i,j\rangle}\sigma_i\sigma_j, \tag{31}$$

where the sum goes over nearest neighbor spin pairs.

Two parallel spins contribute an energy of $-J$ while two antiparallel ones contribute $+J$. In the ferromagnetic case the state of lowest energy is the fully polarized state where all spins are aligned, either pointing up or down.

At finite temperatures the spins start to fluctuate and also states of higher energy contribute to thermal averages. The average magnetization thus decreases from its full value at zero temperature. At a critical temperature T_c there is a second order phase transition to a disordered phase. The Ising model is the simplest magnetic model exhibiting such a phase transition and is often used as a prototype model for magnetism.

The thermal average of a quantity A at a finite temperature T is given by a sum over all states:

$$\langle A \rangle = \frac{1}{Z} \sum_i A_i \exp(-\beta E_i), \tag{32}$$

where $\beta = 1/k_B T$ is the inverse temperature. A_i is the value of the quantity A in the configuration i. E_i is the energy of that configuration.

The partition function

$$Z = \sum_i \exp(-\beta E_i) \tag{33}$$

normalizes the probabilities $p_i = \exp(-\beta E_i)/Z$.

For small systems it is possible to evaluate these sums exactly. As the number of states grows like 2^N a straight-forward summation is possible only for very small N. For large higher dimensional systems Monte Carlo summation/integration is the method of choice.

3.2 The single spin flip Metropolis algorithm

As was discussed in connection with integration it is usually not efficient to estimate the average (32) using simple sampling. The optimal method is importance sampling, where the states i are not chosen uniformly but with the correct probability p_i, which we can again do using the Metropolis algorithm.

The simplest Monte Carlo algorithm for the Ising model is the single spin flip Metropolis algorithm which defines a Markov chain through phase space.

- Starting with a configuration c_i propose to flip a single spin, leading to a new configuration c'.

- Calculate the energy difference $\Delta E = E[c'] - E[c_i]$ between the configurations c' and c_i.

- If $\Delta E < 0$ the next configuration is $c_{i+1} = c'$
- If $\Delta E > 0$ then $c_{i+1} = c'$ with probability $\exp(-\beta\Delta E)$, otherwise $c_{i+1} = c_i$. We do that by drawing a random number r uniformly distributed in the interval $[0, 1[$ and set $c_{i+1} = c'$ if $r < \exp(-\beta\Delta E)$.
- Measure all the quantities of interest in the new configuration.

This algorithm is ergodic since any configuration can be reached from any other in a finite number of spin flips. It also fulfills the detailed balance condition.

3.3 Systematic errors: boundary and finite size effects

In addition to statistical errors due to the Monte Carlo sampling our simulations suffer from systematic errors due to boundary effects and the finite size of the system.

Unless one wants to study finite systems with open boundaries, boundary effects can be avoided completely by using periodic boundary conditions. The lattice is continued periodically, forming a torus. The left neighbor of the leftmost spin is just the rightmost boundary spin, etc..

Although we can avoid boundary effects, finite size effects remain since now all correlations are periodic with the linear system size as period. Here is how we can treat them:

- *Away from phase transitions* the correlation length ξ is finite and finite size effects are negligible if the linear system size $L \gg \xi$. Usually $L > 6\xi$ is sufficient, but this should be checked for each simulation.
- *In the vicinity of continuous phase transitions* we encounter a problem: the correlation length ξ diverges. Finite size scaling can comes to the rescue and can be used to obtain the critical behavior. A detailed discussion of finite size scaling is beyond the scope of these notes.

3.4 Critical behavior of the Ising model

Close to the phase transition at T_c again scaling laws characterize the behavior of all physical quantities. The average magnetization scales as

$$m(T) = \langle |M|/V \rangle \propto (T_c - T)^\beta, \tag{34}$$

where M is the total magnetization and V the system volume (number of spins).

The magnetic susceptibility $\chi = \frac{dm}{dh}|_{h=0}$ can be calculated from magnetization fluctuations and diverges with the exponent γ:

$$\chi(T) = \frac{\langle M^2/V\rangle - \langle |M|/V\rangle^2}{T} \propto |T_c - T|^{-\gamma}. \tag{35}$$

The correlation length ξ is defined by the asymptotically exponential decay of the two-spin correlations:

$$\langle \sigma_0 \sigma_{\mathbf{r}} \rangle - \langle |m| \rangle^2 \propto \exp(-r/\xi). \tag{36}$$

It is best calculated from the structure factor $S(\mathbf{q})$, defined as the Fourier transform of the correlation function. For small $\mathbf{q}$ the structure factor has a Lorentzian shape:

$$S(\mathbf{q}) = \frac{1}{1 + q^2\xi^2} + \mathrm{O}(q^4). \tag{37}$$

The correlation length diverges as

$$\xi(p) \propto |T - T_c|^{-\nu}. \tag{38}$$

At the critical point the correlation function itself follows a power law:

$$\langle \sigma_0 \sigma_{\mathbf{r}} \rangle \propto r^{-(d-2+\eta)} \tag{39}$$

where $\eta = 2\beta/\nu - d + 2$.

The specific heat $C(T)$ diverges logarithmically in two dimensions:

$$C(T) \propto \ln |T - T_c| \propto |T - T_c|^{-\alpha} \tag{40}$$

and the critical exponent $\alpha = 0$.

A good estimate of T_c is obtained from the Binder cumulant

$$U = 1 - \frac{\langle M^4 \rangle}{3\langle M^2 \rangle^2}, \tag{41}$$

which has a universal value at p_c, also the Binder cumulant has a universal value at T_c. The curves of $U(T)$ for different system sizes L all cross in one point at T_c. This is a consequence of the finite size scaling ansatz:

$$\begin{aligned} \langle M^4 \rangle &= (T - T_c)^{4\beta} u_4((T - T_c)L^{1/\nu}) \\ \langle M^2 \rangle &= (T - T_c)^{2\beta} u_2((T - T_c)L^{1/\nu}). \end{aligned} \tag{42}$$

Thus

$$U(T, L) = 1 - \frac{u_4((T - T_c)L^{1/\nu})}{3u_2((T - T_c)L^{1/\nu})^2}, \tag{43}$$

which for $T = T_c$ is universal and independent of system size L:

$$U(T_c, L) = 1 - \frac{u_4(0)}{3u_2(0)^2} \tag{44}$$

High precision Monte Carlo simulations actually show that not all lines cross exactly at the same point, but that due to higher order corrections to finite size scaling the crossing point moves slightly, proportional to $L^{-1/\nu}$, allowing a high precision estimate of T_c and ν. For details of the determination of critical points and exponents see e.g. Ref [1, 2].

3.5 "Critical slowing down" and cluster Monte Carlo methods

The importance of autocorrelation becomes clear when we wish to simulate the Ising model at low temperatures. The mean magnetization $\langle m \rangle$ is zero on any finite cluster, as there is a degeneracy between a configuration and its spin reversed counterpart. If, however, we start at low temperatures with a configuration with all spins aligned up it will take extremely long time for all spins to be flipped by the single spin flip algorithm. This problem appears as soon as we get close to the critical temperature, where it was observed that the autocorrelation times diverge as

$$\tau \propto [\min(\xi, L)]^z. \tag{45}$$

with a dynamical critical exponents $z \approx 2$ for all local update methods like the single spin flip algorithm.

The reason is that at low temperatures it is very unlikely that even one spin gets flipped, and even more unlikely for a large cluster of spins to be flipped. The solution to this problem in the form of cluster updates was found in 1987 and 1989 by Swendsen and Wang [3] and by Wolff [4]. Instead of flipping single spins they propose to flip big clusters of spins and choose them in a clever way so that the probability of flipping these clusters is large.

3.5.1 Cluster updates

We use the Fortuin-Kastelyn representation of the Ising model, as generalized by Kandel and Domany. The phase space of the Ising model is enlarged by assigning a set $\mathcal{G}$ of possible "graphs" to each configuration C in the set of configurations $\mathcal{C}$. We write the partition function as

$$Z = \sum_{C \in \mathcal{C}} \sum_{G \in \mathcal{G}} W(C, G) \tag{46}$$

where the new weights $W(C,G) > 0$ are chosen such that Z is the partition function of the original model by requiring

$$\sum_{G\in\mathcal{G}} W(C,G) = W(C) := \exp(-\beta E[C]), \tag{47}$$

where $E[C]$ is the energy of the configuration C.

The algorithm now proceeds as follows. First we assign a graph $G \in \mathcal{G}$ to the configuration C, chosen with the correct probability

$$P_C(G) = W(C,G)/W(C). \tag{48}$$

Then we choose a new configuration C' with probability $p[(C,G) \rightarrow (C',G)]$, keeping the graph G fixed; next a new graph G' is chosen

$$C \rightarrow (C,G) \rightarrow (C',G) \rightarrow C' \rightarrow (C',G') \rightarrow \ldots \tag{49}$$

What about detailed balance? The procedure for choosing graphs with probabilities P_G obeys detailed balance trivially. The non-trivial part is the probability of choosing a new configuration C'. There detailed balance requires:

$$W(C,G)p[(C,G) \rightarrow (C',G)] = W(C',G)p[(C',G) \rightarrow (C,G)], \tag{50}$$

which can be fulfilled using either the heat bath algorithm

$$p[(C,G) \rightarrow (C',G)] = \frac{W(C',G)}{W(C,G) + W(C',G)} \tag{51}$$

or by again using the Metropolis algorithm:

$$p[(C,G) \rightarrow (C',G)] = \max(W(C',G)/W(C,G), 1) \tag{52}$$

The algorithm simplifies a lot if we can find a graph mapping such that the graph weights do not depend on the configuration whenever it is nonzero in that configuration. This means, we want the graph weights to be

$$W(C,G) = \Delta(C,G)V(G), \tag{53}$$

where

$$\Delta(C,G) := \left\{ \begin{array}{l} 1 \text{ if } W(C,G) \neq 0, \\ 0 \text{ otherwise.} \end{array} \right. . \tag{54}$$

Then equation (51) simply becomes $p = 1/2$ and equation (52) reduces to $p = 1$ for any configuration C' with $W(C',G) \neq 0$.

Table 1: Local bond weights for the Kandel-Domany representation of the Ising model.

	$c=\uparrow\uparrow$	$c=\downarrow\uparrow$	$c=\uparrow\downarrow$	$c=\downarrow\downarrow$	V(g)
$\Delta(c,\text{discon.})$	1	1	1	1	$e^{-\beta J}$
$\Delta(c,\text{con.})$	1	0	0	1	$e^{\beta J}-e^{-\beta J}$
w(c)	$\exp(\beta J)$	$\exp(-\beta J)$	$\exp(-\beta J)$	$\exp(\beta J)$	

3.5.2 The cluster algorithms for the Ising model

Let us now show how this abstract and general algorithm can be applied to the Ising model. Our graphs will be bond-percolation graphs on the lattice. Spins pointing into the same direction can be connected or disconnected. Spins pointing in opposite directions will always be disconnected. In the Ising model we can write the weights $W(C)$ and $W(C,G)$ as products over all bonds b:

$$W(C) = \prod_b w(C_b) \tag{55}$$

$$W(C,G) = \prod_b w(C_b,G_b) = \prod_b \Delta(C_b,G_b)V(G_b) \tag{56}$$

where the local bond configurations C_b can be one of $\{\uparrow\uparrow,\downarrow\uparrow,\uparrow\downarrow,\downarrow\downarrow\}$

and the local graphs can be "connected" or "disconnected". The graph selection can thus be done locally on each bond.

Table 1 shows the local bond weights $w(c,g)$, $w(c)$, $\Delta(c,g)$ and $V(g)$. It can easily be checked that the sum rule (47) is satisfied.

The probability of a connected bond is $[\exp(\beta J)-\exp(-\beta J)]/\exp(\beta J) = 1-\exp(-2\beta J)$ if two spins are aligned and zero otherwise. These connected bonds group the spins into clusters of aligned spins.

A new configuration C' with the same graph G can differ from C only by flipping clusters of connected spins. Thus the name "cluster algorithms". The clusters can be flipped independently, as the flipping probabilities $p[(C,G)\rightarrow(C',G)]$ are configuration independent constants.

There are two variants of cluster algorithms that can be constructed using the rules derived above.

3.5.3 The Swendsen-Wang algorithm

The Swendsen-Wang or multi-cluster algorithm proceeds as follows:

i) Each bond in the lattice is assigned a label "connected" or "disconnected" according to above rules. Two aligned spins are connected with probability $1 - \exp(-2\beta J)$. Two antiparallel spins are never connected.

ii) Next a cluster labeling algorithm, like the Hoshen-Kopelman algorithm is used to identify clusters of connected spins.

iii) Measurements are performed, using improved estimators discussed in the next section.

iv) Each cluster of spins is flipped with probability 1/2.

3.5.4 The Wolff algorithm

The Swendsen Wang algorithm gets less efficient in dimensions higher than two as the majority of the clusters will be very small ones, and only a few large clusters exist. The Wolff algorithm is similar to the Swendsen-Wang algorithm but builds only one cluster starting from a randomly chosen point. As the probability of this point being on a cluster of size s is proportional to s the Wolff algorithm builds preferebly larger clusters. It works in the following way:

i) Choose a random spin as the initial cluster.

ii) If a neighboring spin is parallel to the initial spin it will be added to the cluster with probability $1 - \exp(-2\beta J)$.

iii) Repeat step ii) for all points newly added to the cluster and repeat this procedure until no new points can be added.

iv) Perform measurements using improved estimators.

v) Flip all spins in the cluster.

We will see in the next section that the linear cluster size diverges with the correlation length ξ and that the average number of spins in a cluster is just χT. Thus the algorithm adapts optimally to the physics of the system and the dynamical exponent $z \approx 0$, thus solving the problem of critical slowing down. Close to criticality these algorithms are many orders of magnitudes (a factor L^2) better than the local update methods.

3.6 Improved Estimators

In this section we present a neat trick that can be used in conjunction with cluster algorithms to reduce the variance, and thus the statistical error of Monte Carlo measurements. Not only do these “improved estimators” reduce the variance. They are also much easier to calculate than the usual “simple estimators”.

To derive them we consider the Swendsen-Wang algorithm. This algorithm divides the lattice into N_c clusters, where all spins within a cluster are aligned. The next possible configuration is any of the 2^{N_c} configurations that can be reached by flipping any subset of the clusters. The idea behind the “improved estimators” is to measure not only in the new configuration but in all equally probable 2^{N_c} configurations.

As simplest example we consider the average magnetization $\langle m\rangle$. We can measure it as the expectation value $\langle\sigma_{\vec{i}}\rangle$ of a single spin. As the cluster to which the spin belongs can be freely flipped, and the flipped cluster has the same probability as the original one, the improved estimator is

$$\langle m\rangle = \langle\frac{1}{2}(\sigma_{\vec{i}} - \sigma_{\vec{i}})\rangle = 0. \tag{57}$$

This result is obvious because of symmetry, but we saw that at low temperatures a single spin flip algorithm will fail to give this correct result since it takes an enormous time to flip all spins. Thus it is encouraging that the cluster algorithms automatically give the exact result in this case.

Correlation functions are not much harder to measure:

$$\langle\sigma_{\vec{i}}\sigma_{\vec{j}}\rangle = \begin{cases} 1 & \text{if } \vec{i} \text{ und } \vec{j} \text{ are on the same cluster} \\ 0 & \text{otherwise} \end{cases} \tag{58}$$

To derive this result consider the two cases and write down the improved estimators by considering all possible cluster flips.

Using this simple result for the correlation functions the mean square of the magnetization is

$$\langle m^2\rangle = \frac{1}{N^2}\sum_{\vec{i},\vec{j}}\langle\sigma_{\vec{i}}\sigma_{\vec{j}}\rangle = \frac{1}{N^2}\langle\sum_{cluster} S(cluster)^2\rangle, \tag{59}$$

where $S(cluster)$ is the number of spins in a cluster. The susceptibility above T_c is simply given by $\beta\langle m^2\rangle$ and can also easily be calculated by above sum over the squares of the cluster sizes.

In the Wolff algorithm only a single cluster is built. Above sum (59) can be rewritten to be useful also in case of the Wolff algorithm:

$$\langle m^2\rangle \quad = \quad \frac{1}{N^2}\langle\sum_{cluster} S(cluster)^2\rangle$$

$$
\begin{aligned}
&= \frac{1}{N^2}\sum_{\vec{i}} \frac{1}{S_{\vec{i}}} S_{\vec{i}}^2 \\
&= \frac{1}{N^2}\sum_{\vec{i}} S_{\vec{i}} = \frac{1}{N}\langle S(\text{cluster})\rangle, \qquad (60)
\end{aligned}
$$

where $S_{\vec{i}}$ is the size of the cluster containing the initial site $\vec{i}$. The expectation value for m^2 is thus simply the mean cluster size. In this derivation we replaced the sum over all clusters by a sum over all sites and had to divide the contribution of each cluster by the number of sites in the cluster. Next we can replace the average over all lattice sites by the expectation value for the cluster on a randomly chosen site, which in the Wolff algorithm will be just the one Wolff cluster we build.

Generalizations to other quantities, like the structure factor $S(\vec{q})$ are straightforward. While the calculation of $S(\vec{q})$ by Fourier transform needs at least $O(N \ln N)$ steps, it can be done much faster using improved estimators, here derived for the Wolff algorithm:

$$
\begin{aligned}
\langle S(\vec{q})\rangle &= \frac{1}{N^2}\sum_{\vec{r},\vec{r}'} \sigma_{\vec{r}}\sigma_{\vec{r}'} \exp(i\vec{q}(\vec{r}-\vec{r}')) \\
&= \frac{1}{NS(cluster)} \sum_{\vec{r},\vec{r}' \in cluster} \sigma_{\vec{r}}\sigma_{\vec{r}'} \exp(i\vec{q}(\vec{r}-\vec{r}')) \\
&= \frac{1}{NS(cluster)} \left| \sum_{\vec{r} \in cluster} \exp(i\vec{q}\vec{r}) \right|^2, \qquad (61)
\end{aligned}
$$

This needs only $O(S(cluster))$ operations and can be measured directly when constructing the cluster.

Care must be taken for higher order correlation functions. Improved estimators for quantities like m^4 which need at least two clusters and cannot be measured in an improved way using the Wolff algorithm.

3.7 Generalizations of cluster algorithms

Cluster algorithms can be used not only for the Ising model but for a large class of classical, and even quantum spin models. The quantum version is the “loop algorithm”, which will be discussed later in the course. In this section we discuss generalizations to other classical spin models.

Before discussing specific models we remark that generalizations to models with different coupling constants on different bonds, or even random couplings are straightforward. All decisions are done locally, individually for each spin or bond, and the couplings can thus be different at each bond.

3.7.1 Potts models

q-state Potts models are the generalization of the Ising model to more than two states. The Hamilton function is

$$H = -J \sum_{\langle i,j \rangle} \delta_{s_i,s_j}, \tag{62}$$

where the states s_i can take any integer value in the range $1, \ldots, q$. The 2-state Potts model is just the Ising model with some trivial rescaling.

The cluster algorithms for the Potts models connect spins with probability $1 - e^{-\beta J}$ if the spins have the same value. The clusters are then "flipped" to any arbitrarily chosen value in the range $1, \ldots, q$.

3.7.2 $O(N)$ models

Another, even more important generalization are the $O(N)$ models. Well known examples are the XY-model with $N = 2$ and the Heisenberg model with $N = 3$. In contrast to the Ising model the spins can point into any arbitrary direction on the N-sphere. The spins in the XY model can point into any direction in the plane and can be characterized by a phase. The spins in the Heisenberg model point into any direction on a sphere.

The Hamilton function is:

$$H = -J \sum_{\langle i,j \rangle} \vec{S}_i \vec{S}_j, \tag{63}$$

where the states $\vec{S}_i$ are $SO(N)$ vectors.

Cluster algorithms are constructed by projecting all spins onto a random direction $\vec{e} \in SO(N)$. The cluster algorithm for the Ising model can then be used for this projection. Two spins $\vec{S}_i$ and $\vec{S}_j$ are connected with probability

$$1 - \exp\Big(\min[0, -2\beta J(\vec{e} \cdot \vec{S}_i)(\vec{e} \cdot \vec{S}_j)]\Big)\,. \tag{64}$$

The spins are flipped by inverting the projection onto the $\vec{e}$-direction:

$$\vec{S}_i \to \vec{S}_i - 2(\vec{e} \cdot \vec{S}_i)\vec{e}. \tag{65}$$

In the next update step a new direction $\vec{e}$ is chosen.

4 The quantum Monte Carlo loop algorithm

The "loop-algorithm" is a generalization of the classical Swendsen-Wang cluster algorithms to quantum lattice models. I will discuss it here in a path-integral representation. A slightly modified version will be discussed later in the context of the SSE representation.

4.1 Path-integral representation in terms of world lines

I will discuss the loop algorithm for a spin-1/2 quantum XXZ model with the Hamiltonian

$$\begin{aligned} H &= -\sum_{\langle i,j\rangle}\left(J_z S_i^z S_j^z + J_{xy}(S_i^x S_j^x + S_i^y S_j^y)\right) \\ &= -\sum_{\langle i,j\rangle}\left(J_z S_i^z S_j^z + \frac{J_{xy}}{2}(S_i^+ S_j^- + S_i^- S_j^+)\right). \end{aligned} \tag{66}$$

For $J \equiv J_z = J_{xy}$ we have the Heisenberg model ($J > 0$ is ferromagnetic, $J < 0$ antiferromagnetic). $J_{xy} = 0$ is the (classical) Ising model and $J_z = 0$ the quantum XY model.

In the quantum Monte Carlo simulation we want to evaluate thermodynamic averages such as

$$\langle A\rangle = \frac{\mathrm{Tr} A e^{-\beta H}}{\mathrm{Tr} e^{-\beta H}}. \tag{67}$$

The main problem is the calculation of the exponential $e^{-\beta H}$. The straightforward calculation would require a complete diagonalization, which is just what we want to avoid. We thus discretize the imaginary time (inverse temperature) direction[1] and subdivide $\beta = M\Delta\tau$:

$$e^{-\beta H} = \left(e^{-\Delta\tau H}\right)^M = (1-\Delta\tau H)^M + O(\Delta\tau) \tag{68}$$

In the limit $M \to \infty$ ($\Delta\tau \to 0$) this becomes exact. We will take the limit later, but stay at finite $\Delta\tau$ for now.

The next step is to insert the identity matrix, represented by a sum over all basis states $1 = \sum_i |i\rangle\langle i|$ between all operators $(1-\Delta\tau H)$:

$$\begin{aligned} Z &= \mathrm{Tr} e^{-\beta H} = \mathrm{Tr}\,(1-\Delta\tau H)^M + O(\Delta\tau) \\ &= \sum_{i_1,\ldots,i_M} \langle i_1|1-\Delta\tau H|i_2\rangle\langle i_2|1-\Delta\tau H|i_3\rangle\cdots\langle i_M|1-\Delta\tau H|i_1\rangle + O(\Delta\tau) \\ &=: P_{i_1,\ldots,i_M} \end{aligned} \tag{69}$$

and similarly for the measurement, obtaining

$$\langle A\rangle = \sum_{i_1,\ldots,i_M} \frac{\langle i_1|A(1-\Delta\tau H)|i_2\rangle}{\langle i_1|1-\Delta\tau H|i_2\rangle} P_{i_1,\ldots,i_M} + O(\Delta\tau). \tag{70}$$

[1] Time evolution in quantum mechanics is e^{-itH}. The Boltzman factor $e^{-\beta H}$ thus corresponds to an evolution in imaginary time $t = -i\beta$

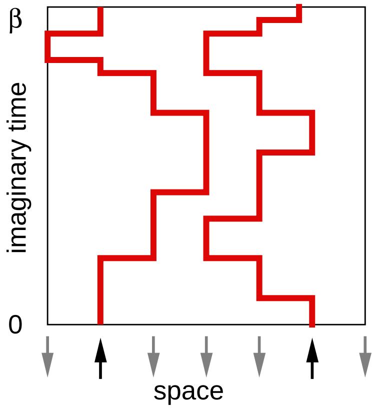


Figure 1: Example of a world line configuration for a spin-1/2 quantum Heisenberg model. Drawn are the world lines for up-spins only. Down spin world lines occupy the rest of the configuration.

If we choose the basis states $|i\rangle$ to be eigenstates of the local S^z operators we end up with an Ising-like spin system in one higher dimension. Each choice $i_1, \ldots, i_M$ corresponds to one of the possible configurations of this classical spin system. The trace is mapped to periodic boundary conditions in the imaginary time direction of this classical spin system. The probabilities are given by matrix elements $\langle i_n|1-\Delta\tau H|i_{n+1}\rangle$. We can now sample this classical system using classical Monte Carlo methods.

However, most of the matrix elements $\langle i_n|1 - \Delta\tau H|i_{n+1}\rangle$ are zero, and thus nearly all configurations have vanishing weight. The only non-zero configurations are those where neighboring states $|i_n\rangle$ and $|i_{n+1}\rangle$ are either equal or differ by one of the off-diagonal matrix elements in H, which are nearest neighbor exchanges by two opposite spins. We can thus uniquely connect spins on neighboring "time slices" and end up with world lines of the spins, sketched in Fig. 1. Instead of sampling over all configurations of local spins we thus have to sample only over all world line configurations (the others have vanishing weight). Our update moves are not allowed to break world lines but have to lead to new valid world line configurations.

4.2 The loop algorithm

Until 1993 only local updates were used, which suffered from a slowing down like in the classical case. The solution came as a generalization of the cluster algorithms to quantum systems [5, 6].

This algorithm is best described by first taking the continuous time limit

Table 2: The six local configurations for an XXZ model and their weights.

configuration	weight
$S_i(\tau+d\tau)\uparrow \quad \uparrow S_j(\tau+d\tau)$ $S_i(\tau)\uparrow \quad \uparrow S_j(\tau)$, $S_i(\tau+d\tau)\downarrow \quad \downarrow S_j(\tau+d\tau)$ $S_i(\tau)\downarrow \quad \downarrow S_j(\tau)$	$1+\frac{J_z}{4}d\tau$
$S_i(\tau+d\tau)\uparrow \quad \downarrow S_j(\tau+d\tau)$ $S_i(\tau)\uparrow \quad \downarrow S_j(\tau)$, $S_i(\tau+d\tau)\downarrow \quad \uparrow S_j(\tau+d\tau)$ $S_i(\tau)\downarrow \quad \uparrow S_j(\tau)$	$1-\frac{J_z}{4}d\tau$
$S_i(\tau+d\tau)\downarrow \quad \uparrow S_j(\tau+d\tau)$ $S_i(\tau)\uparrow \quad \downarrow S_j(\tau)$, $S_i(\tau+d\tau)\uparrow \quad \downarrow S_j(\tau+d\tau)$ $S_i(\tau)\downarrow \quad \uparrow S_j(\tau)$	$\frac{J_{xy}}{2}d\tau$

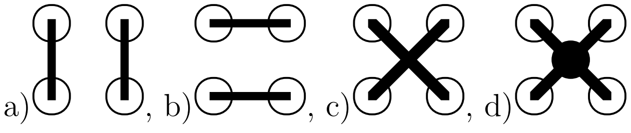


Figure 2: The four local graphs: a) vertical, b) horizontal c) crossing and d) freezing (connects all four corners).

$M \to \infty$ $(\Delta\tau \to d\tau)$ and by working with infinitesimals. Similar to the Ising model we look at two spins on neigboring sites i and j at two neighboring times τ and $\tau + d\tau$, as sketched in Table 2. There are a total of six possible configurations, having three different probabilities. The total probabilities are the products of all local probabilities, like in the classical case. This is obvious for different time slices. For the same time slice it is also true since, denoting by H_{ij} the term in the Hamiltonian H acting on the bond between sites i and j we have $\prod_{\langle i,j\rangle}(1 - d\tau H_{ij}) = 1 - d\tau \sum_{\langle i,j\rangle} H_{ij} = 1 - d\tau H$. In the following we focus only on such local four-spin plaquettes. Next we again use the Kandel-Domany framework and assign graphs. As the updates are not allowed to break world lines only four graphs, sketched in Fig. 2 are allowed. Finally we have to find Δ functions and graph weights that give the correct probabilities. The solution for the XY-model, ferromagnetic and antiferromagnetic Heisenberg model and the Ising model is shown in Tables 3 - 6.

Let us first look at the special case of the Ising model. As the exchange term is absent in the Ising model all world lines run straight and can be replaced by classical spins. The only non-trivial graph is the "freezing", connecting two neighboring world lines. Integrating the probability that two neighboring sites are *nowhere* connected along the time direction we obtain:

Table 3: The graph weights for the quantum-XY model and the Δ function specifying whether the graph is allowed. The dash – denotes a graph that is not possible for a configuration because of spin conservation and has to be zero.

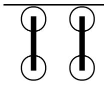
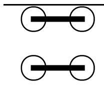
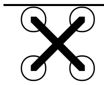
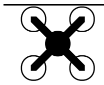
G	$\Delta\left(\begin{smallmatrix}\uparrow&\uparrow\\\uparrow&\uparrow\end{smallmatrix}, G\right)$ $= \Delta\left(\begin{smallmatrix}\downarrow&\downarrow\\\downarrow&\downarrow\end{smallmatrix}, G\right)$	$\Delta\left(\begin{smallmatrix}\uparrow&\downarrow\\\uparrow&\downarrow\end{smallmatrix}, G\right)$ $= \Delta\left(\begin{smallmatrix}\downarrow&\uparrow\\\downarrow&\uparrow\end{smallmatrix}, G\right)$	$\Delta\left(\begin{smallmatrix}\downarrow&\uparrow\\\uparrow&\downarrow\end{smallmatrix}, G\right)$ $= \Delta\left(\begin{smallmatrix}\uparrow&\downarrow\\\downarrow&\uparrow\end{smallmatrix}, G\right)$	graph weight
	1	1	–	$1 - \frac{J_{xy}}{4} d\tau$
	–	1	1	$\frac{J_{xy}}{4} d\tau$
	1	–	1	$\frac{J_{xy}}{4} d\tau$
	0	0	0	0
total weight	1	1	$\frac{J_{xy}}{2} d\tau$	

Table 4: The graph weights for the ferromagnetic quantum Heisenberg model and the Δ function specifying whether the graph is allowed. The dash – denotes a graph that is not possible for a configuration because of spin conservation and has to be zero.

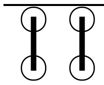
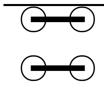
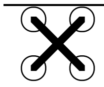
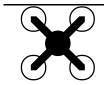
G	$\Delta\left(\begin{smallmatrix}\uparrow&\uparrow\\\uparrow&\uparrow\end{smallmatrix}, G\right)$ $= \Delta\left(\begin{smallmatrix}\downarrow&\downarrow\\\downarrow&\downarrow\end{smallmatrix}, G\right)$	$\Delta\left(\begin{smallmatrix}\uparrow&\downarrow\\\uparrow&\downarrow\end{smallmatrix}, G\right)$ $= \Delta\left(\begin{smallmatrix}\downarrow&\uparrow\\\downarrow&\uparrow\end{smallmatrix}, G\right)$	$\Delta\left(\begin{smallmatrix}\downarrow&\uparrow\\\uparrow&\downarrow\end{smallmatrix}, G\right)$ $= \Delta\left(\begin{smallmatrix}\uparrow&\downarrow\\\downarrow&\uparrow\end{smallmatrix}, G\right)$	graph weight
	1	1	–	$1 - \frac{J}{4} d\tau$
	–	0	0	0
	1	–	1	$\frac{J}{2} d\tau$
	0	0	0	0
total weight	$1 + \frac{J}{4} d\tau$	$1 - \frac{J}{4} d\tau$	$\frac{J}{2} d\tau$	

Table 5: The graph weights for the antiferromagnetic quantum Heisenberg model and the Δ function specifying whether the graph is allowed. The dash – denotes a graph that is not possible for a configuration because of spin conservation and has to be zero. To avoid the sign problem (see next subsection) we change the sign of J_{xy}, which is allowed only on bipartite lattices.

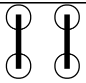
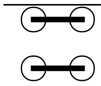
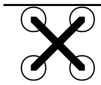
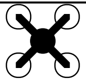
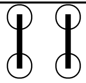
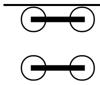
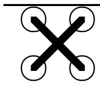
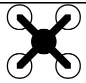
G	$\Delta(\overset{\uparrow}{\uparrow}\ \overset{\uparrow}{\uparrow}, G)$ $= \Delta(\overset{\downarrow}{\downarrow}\ \overset{\downarrow}{\downarrow}, G)$	$\Delta(\overset{\uparrow}{\uparrow}\ \overset{\downarrow}{\downarrow}, G)$ $= \Delta(\overset{\downarrow}{\downarrow}\ \overset{\uparrow}{\uparrow}, G)$	$\Delta(\overset{\downarrow}{\uparrow}\ \overset{\uparrow}{\downarrow}, G)$ $= \Delta(\overset{\uparrow}{\downarrow}\ \overset{\downarrow}{\uparrow}, G)$	graph weight
	1	1	–	$1 - \frac{\lvert J\rvert}{4} d\tau$
	–	1	1	$\frac{\lvert J\rvert}{2} d\tau$
	0	–	0	0
	0	0	0	0
total weight	$1 - \frac{\lvert J\rvert}{4} d\tau$	$1 + \frac{\lvert J\rvert}{4} d\tau$	$\frac{\lvert J\rvert}{2} d\tau$	

Table 6: The graph weights for the ferromagnetic Ising model and the Δ function specifying whether the graph is allowed. The dash – denotes a graph that is not possible for a configuration because of spin conservation and has to be zero.

G	$\Delta(\overset{\uparrow\ \uparrow}{\uparrow\ \uparrow}, G)$ $= \Delta(\overset{\downarrow\ \downarrow}{\downarrow\ \downarrow}, G)$	$\Delta(\overset{\uparrow\ \downarrow}{\uparrow\ \downarrow}, G)$ $= \Delta(\overset{\downarrow\ \uparrow}{\downarrow\ \uparrow}, G)$	$\Delta(\overset{\downarrow\ \uparrow}{\uparrow\ \downarrow}, G)$ $= \Delta(\overset{\uparrow\ \downarrow}{\downarrow\ \uparrow}, G)$	graph weight
	1	1	–	$1 - \frac{J_z}{4} d\tau$
	–	0	0	0
	0	–	0	0
	1	0	0	$\frac{J_z}{2} d\tau$
total weight	$1 + \frac{J_z}{4} d\tau$	$1 - \frac{J_z}{4} d\tau$	0	

times:

$$\prod_{\tau=0}^{\beta} (1 - d\tau J/2) = \lim_{M\to\infty} (1 - \Delta\tau J/2)^M = \exp(-\beta J/2) \qquad (71)$$

Taking into account that the spin is $S = 1/2$ and the corresponding classical coupling $J_{cl} = S^2 J = J/4$ we find for the probability that two spins are *connected*: $1 - \exp(-2\beta J_{cl})$. We end up exactly with the cluster algorithm for the classical Ising model!

The other cases are special. Here each graph connects two spins. As each of these spins is again connected to only one other, all spins connected by a cluster form a closed loop, hence the name "loop algorithm". Only one issue remains to be explained: how do we assign a horizontal or crossing graph with infinitesimal probability, such as $(J/2)d\tau$. This is easily done by comparing the assignment process with radioactive decay. For each segment the graph runs vertical, except for occasional decay processes occuring with probability $(J/2)d\tau$. Instead of asking at every infinitesimal time step whether a decay occurs we simply calculate an exponentially distributed decay time t using an exponential distribution with decay constant $J/2$. Looking up the equation in the lecture notes of the winter semester we have $t = -(2/J)\ln(1-u)$ where u is a uniformly distributed random number.

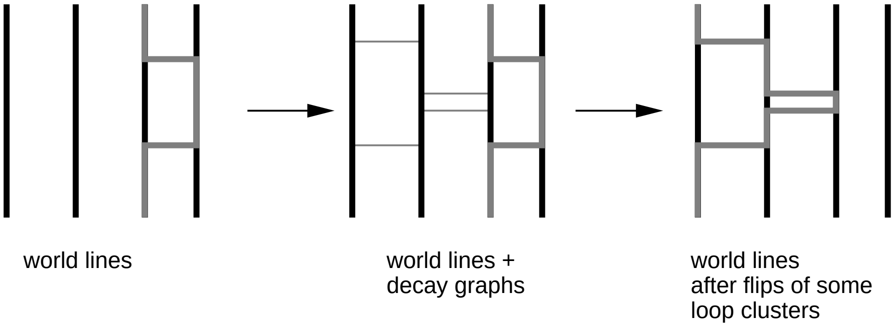


Figure 3: Example of a loop update. In a first step decay paths are inserted where possible at positions drawn randomly according to an exponential distribution and graphs are assigned to all exchange terms (hoppings of world lines). In a second stage (not shown) the loop clusters are identified. Finally each loop cluster is flipped with probability 1/2 and one ends up with a new configuration.

The algorithm now proceeds as follows (see Fig. 3): for each bond we start at time 0 and calculate a decay time. If the spins at that time are oriented properly and an exchange graph is possible we insert one. Next we advance by another randomly chosen decay time along the same bond and repeat the procedure until we have reached the extent β. This assigns graphs to all infinitesimal time steps where spins do not change. Next we assign a graph to all of the (finite number of) time steps where two spins are exchanged. In the case of the Heisenberg models there is always only one possible graph to assign and this is very easy. In the next step we identify the loop-clusters and then flip them each with probability 1/2. Alternatively a Wolff-like algorithm can be constructed that only builds one loop-cluster.

Improved estimators for measurements can be constructed like in classical models. The derivation is similar to the classical models. I will just mention two simple ones for the ferromagnetic Heisenberg model. The spin-spin corelation is

$$S_i^z(\tau)S_j^z(\tau') = \begin{cases} 1 & \text{if } (i,\tau) \text{ und } (j,\tau') \text{ are on the same cluster} \\ 0 & \text{otherwise} \end{cases} \tag{72}$$

and the uniform susceptibilty is

$$\chi = \frac{1}{N\beta}\sum_c S(c)^2, \tag{73}$$

where the sum goes over all loop clusters and $S(c)$ is the length of all the loop segments in the loop cluster c.

For further information on the loop algorithm I refer to the recent review by Evertz [7].

4.3 The negative sign problem

Now that we have algorithms with no critical slowing down we could think that we have completely solved the problem of quantum many body problems.

There is however the *negative sign problem* which destroys our dreams. We need to interpret the matrix elements $\langle i_n|1-\Delta\tau H|i_{n+1}\rangle$ as probablities, which requires them to be positive. However all off-diagonal positive matrix elements of H arising from Fermi statistics give rise to a negative probability in fermionic systems and in frustrated quantum magnets.

The simplest example is the exchange term $-(J_{xy}/2)(S_i^+S_j^- + S_i^-S_j^+)$ in the Hamiltonian (66) in the case of an antiferromagnet with $J_{xy} < 0$. For any bipartite lattice, such as chains, square lattices or cubic lattices with there is always an even number of such exchanges and we get rescued once more. For non-bipartite lattices (such as a triangular lattice), on which the antiferromagnetic order is frustrated there is no way around the sign problem. Similarly a minus sign occurs in all configurations where two fermions are exchanged.

Even when there is a sign problem we can still do a simulation. Instead of sampling

$$\langle A\rangle_p := \frac{\int A(x)p(x)dx}{\int p(x)dx} \tag{74}$$

we rewrite this equation as [8]

$$\langle A\rangle_p = \frac{\frac{\int A(x)\mathrm{sign}(p(x))|p(x)|dx}{\int |p(x)|dx}}{\frac{\int \mathrm{sign}(p(x))|p(x)|dx}{\int |p(x)|dx}} = \frac{\langle A\cdot \mathrm{sign}p\rangle_{|p|}}{\langle \mathrm{sign}p\rangle_{|p|}}. \tag{75}$$

We sample with the absolute values $|p|$ and include the sign in the observable. The "sign problem" is the fact that the errors get blown up by an additional factor $1/\langle \mathrm{sign}p\rangle_{|p|}$, which grows exponentially with volume and inverse temperature β, as $\langle \mathrm{sign}p\rangle_{|p|} \propto \exp(-\mathrm{const}\times\beta N)$. Then we are unfortunately back to exponential scaling. Many people have tried to solve the sign problem using basis changes or clever reformulations, but – except for special cases – nobody has succeeded yet. In fact we could show that in some cases the sign problem is **NP**-hard and a solution thus all but impossible [9].

If you want you can try your luck: the person who finds a general solution to the sign problem will surely get a Nobel prize!

5 Exact diagonalization

5.1 Creating the basis set and matrix

The most accurate method for quantum lattice models is exact diagonalization of the Hamiltonian matrix using the Lanczos algorithm. The size of the Hilbert space of an N-site system [4^N for a Hubbard model , 3^N for a t-J model and $(2S+1)^N$ for a spin-S model] can be reduced by making use of symmetries. Translational symmetries can be employed by using Bloch waves with fixed momentum as basis states. Conservation of particle number and spin allows to restrict a calculation to subspaces of fixed particle number and magnetization.

As an example I will sketch how to implement exact diagonalization for a simple one-dimensional spinless fermion model with nearest neighbor hopping t and nearest neighbor repulsion V:

$$H = -t \sum_{i=1}^{L-1} (c_i^\dagger c_{i+1} + \text{H.c.}) + V \sum_{i}^{L-1} n_i n_{i+1}. \tag{76}$$

The first step is to construct a basis set. We describe a basis state as an unsigned integer where bit i set to one corresponds to an occupied site i. As the Hamiltonian conserves the total particle number we thus want to construct a basis of all states with N particles on L sites (or N bits set to one in L bits). The function `state(i)` returns the state corresponding to the i-th basis state, and the function `index`(`s`) returns the number of a basis state `s`.

```
#include <vector>
#include <alps/bitops.h>

class FermionBasis {
public:
  typedef unsigned int state_type;
  typedef unsigned int index_type;
  FermionBasis (int L, int N);

  state_type state(index_type i) const {return states_[i];}
  index_type index(state_type s) const {return index_[s];}
  unsigned int dimension() const { return states_.size();}

private:
  std::vector<state_type> states_;
  std::vector<index_type> index_;
};


FermionBasis::FermionBasis(int L, int N)
{
  index_.resize(1<<L); // 2^L entries
  for (state_type s=0;s<index_.size();++s)
    if(alps::popcnt(s)==N) {
      // correct number of particles
      states_.push_back(s);
      index_[s]=states_.size()-1;
    }
    else
      // invalid state
      index_[s]=std::numeric_limits<index_type>::max();
}
```

Next we have to implement a matrix-vector multiplication $v = Hw$ for the Hamiltonian:

```
class HamiltonianMatrix : public FermionBasis {
public:
  HamiltonianMultiplier(int L, int N, double t, double V)
   : FermionBasis(L,N), t_(t), V_(V), L_(L) {}

  void multiply(std::valarray<double>& v, const std::valarray<double>& w);

private:
  double t_, V_;
  int L_;
}

void HamiltonianMatrix::multiply(std::valarray<double>& v,
               const std::valarray<double>& w)
{
  // do the V-term
  for (int i=0;i<dimension();++i)
  {
    state_type s = state(i);
    // count number of neighboring fermion pairs
    v[i]=w[i]*V_*alps::popcnt(s&(s>>1));
  }

  // do the t-term
  for (int i=0;i<dimension();++i)
  {
    state_type s = state(i);
    for (int r=0;r<L_-1;++r) {
      state_type shop = s^(3<<r);  // exchange two particles
      index_type idx = index(shop); // get the index
      if(idx!=std::numeric_limits<index_type>::max())
        v[idx]+=w[i]*t;
    }
  }
}
```

This class can now be used together with the Lanczos algorithm to calculate the energies and wave functions of the low lying states of the Hamiltonian.

5.2 The Lanczos algorithm

Sparse matrices with only $O(N)$ non-zero elements are very common in scientific simulations. We have already encountered them in the winter semester when we discretized partial differential equations. Now we have reduced the transfer matrix of the Ising model to a sparse matrix product. We will later see that also the quantum mechanical Hamilton operators in lattice models are sparse.

The importance of sparsity becomes obvious when considering the cost of matrix operations as listed in table 7. For large N the sparsity leads to memory and time savings of several orders of magnitude.

Here we will discuss the iterative calculation of a few of the extreme eigenvalues of a matrix by the Lanczos algorithm. Similar methods can be used to solve sparse linear systems of equations.

To motivate the Lanczos algorithms we will first take a look at the power method for a matrix A. Starting from a random initial vector u_1 we calculate

Table 7: Time and memory complexity for operations on sparse and dense $N \times N$ matrices

operation	time	memory
storage		
dense matrix	—	N^2
sparse matrix	—	$\mathrm{O}(N)$
matrix-vector multiplication		
dense matrix	$\mathrm{O}(N^2)$	$\mathrm{O}(N^2)$
sparse matrix	$\mathrm{O}(N)$	$\mathrm{O}(N)$
matrix-matrix multiplication		
dense matrix	$\mathrm{O}(N^{\log 7/\log 2})$	$\mathrm{O}(N^2)$
sparse matrix	$\mathrm{O}(N) \ldots \mathrm{O}(N^2)$	$\mathrm{O}(N) \ldots \mathrm{O}(N^2)$
all eigen values and vectors		
dense matrix	$\mathrm{O}(N^3)$	$\mathrm{O}(N^2)$
sparse matrix (iterative)	$\mathrm{O}(N^2)$	$\mathrm{O}(N^2)$
some eigen values and vectors		
dense matrix (iterative)	$\mathrm{O}(N^2)$	$\mathrm{O}(N^2)$
sparse matrix (iterative)	$\mathrm{O}(N)$	$\mathrm{O}(N)$

the sequence

$$u_{n+1} = \frac{Au_n}{||Au_n||}, \tag{77}$$

which converges to the eigenvector of the largest eigenvalue of the matrix A. The Lanczos algorithm optimizes this crude power method.

5.2.1 Lanczos iterations

The Lanczos algorithm builds a basis $\{v_1, v_2, \ldots, v_M\}$ for the Krylov-subspace $K_M = span\{u_1, u_2, \ldots, u_M\}$, which is constructed by M iterations of equation (77). This is done by the following iterations:

$$\beta_{n+1} v_{n+1} = Av_n - \alpha_n v_n - \beta_n v_{n-1}, \tag{78}$$

where

$$\alpha_n = v_n^\dagger A v_n, \qquad \beta_n = |v_n^\dagger A v_{n-1}|. \tag{79}$$

As the orthogonality condition

$$v_i^\dagger v_j = \delta_{ij} \tag{80}$$

does not determine the phases of the basis vectors, the β_i can be chosen to be real and positive. As can be seen, we only need to keep three vectors of size N in memory, which makes the Lanczos algorithm very efficient, when compared to dense matrix eigensolvers which require storage of order N^2.

In the Krylov basis the matrix A is tridiagonal

$$T^{(n)} := \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \beta_2 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_n \\ 0 & \cdots & 0 & \beta_n & \alpha_n \end{bmatrix}. \tag{81}$$

The eigenvalues $\{\tau_1, \ldots, \tau_M\}$ of T are good approximations of the eigenvalues of A. The extreme eigenvalues converge very fast. Thus $M \ll N$ iterations are sufficient to obtain the extreme eigenvalues.

5.2.2 Eigenvectors

It is no problem to compute the eigenvectors of T. They are however given in the Krylov basis $\{v_1, v_2, \ldots, v_M\}$. To obtain the eigenvectors in the original basis we need to perform a basis transformation.

Due to memory constraints we usually do not store all the v_i, but only the last three vectors. To transform the eigenvector to the original basis we have to do the Lanczos iterations a second time. Starting from the same initial vector v_1 we construct the vectors v_i iteratively and perform the basis transformation as we go along.

5.2.3 Roundoff errors and ghosts

In exact arithmetic the vectors $\{v_i\}$ are orthogonal and the Lanczos iterations stop after at most $N-1$ steps. The eigenvalues of T are then the exact eigenvalues of A.

Roundoff errors in finite precision cause a loss of orthogonality. There are two ways to deal with that:

- Reorthogonalization of the vectors after every step. This requires storing all of the vectors $\{v_i\}$ and is memory intensive.

- Control of the effects of roundoff.

We will discuss the second solution as it is faster and needs less memory. The main effect of roundoff errors is that the matrix T contains extra spurious eigenvalues, called "ghosts". These ghosts are not real eigenvalues of A. However they converge towards real eigenvalues of A over time and increase their multiplicities.

A simple criterion distinguishes ghosts from real eigenvalues. Ghosts are caused by roundoff errors. Thus they do not depend on on the starting vector v_1. As a consequence these ghosts are also eigenvalues of the matrix $\tilde{T}$, which can be obtained from T by deleting the first row and column:

$$\tilde{T}^{(n)} := \begin{bmatrix} \alpha_2 & \beta_3 & 0 & \cdots & 0 \\ \beta_3 & \alpha_3 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_n \\ 0 & \cdots & 0 & \beta_n & \alpha_n \end{bmatrix}. \tag{82}$$

From these arguments we derive the following heuristic criterion to distinguish ghosts from real eigenvalues:

- All multiple eigenvalues are real, but their multiplicities might be too large.
- All single eigenvalues of T which are *not* eigenvalues of $\tilde{T}$ are also real.

Numerically stable and efficient implementations of the Lanczos algorithm can be obtained from netlib. As usual, do not start coding your own algorithm but use existing optimal implementations.

References

[1] A. M. Ferrenberg and D. P. Landau, Phys. Rev. B **44** 5081 (1991).

[2] K. Chen, A. M. Ferrenberg and D. P. Landau, Phys. Rev. B **48** 3249 (1993).

[3] R.H. Swendsen and J-S. Wang, Phys. Rev. Lett. **58**, 86 (1987).

[4] U. Wolff, Phys. Rev. Lett. **62**, 361 (1989).

[5] H. G. Evertz et al., Phys. Rev. Lett. **70**, 875 (1993).

[6] B. B. Beard and U.-J. Wiese, Phys. Rev. Lett. **77**, 5130 (1996).

[7] H.G. Evertz, Adv. Phys. **52**, 1 (2003).

[8] J. E. Hirsch, R. L. Sugar, D. J. Scalapino and R. Blankenbecler, Phys. Rev. B **26**, 5033 (1982).

[9] M. Troyer and U.-J. Wiese, preprint.