Simulations with the Hybrid Monte Carlo algorithm: implementation and data analysis

Stefan Schaefer
Humboldt Universität zu Berlin, Institut für Physik, Newtonstr. 15, 12489 Berlin, Germany

Contents

1
Introduction

The Hybrid Monte Carlo (HMC) algorithm(Duane *et al.*, 1987), together with its variants, is currently the most popular method for simulations of full QCD on the lattice. It is an exact algorithm, which can be used for virtually any theory with continuous variables. It is a method of choice when many degrees of freedom are coupled and single variable updates are not feasible.

Because modern QCD packages are rather involved, the idea behind the course documented here, was to write an HMC code for ϕ^4 theory in D dimensions from scratch and then perform some simulations. This was targeted at students who do not work daily on the numerical aspects of lattice field theory and want to gain familiarity with numerical computations, data analysis and the language used to discuss it. The mathematical background of Markov chain Monte Carlo and the HMC were introduced in the lectures by M. Lüscher at this school(Lüscher, 2010), however, for consistency, the basic definitions are given below. Whereas writing a full code for SU(3) gauge theory would have been beyond the scope of this course, a simple theory like ϕ^4 provides a feasible object for the basic ideas behind such a simulation and, apart from the complications concerning the group variables, the ϕ^4 code proceeds exactly as one for the gauge theory.

Studying ϕ^4 theory numerically has a long tradition in field theory. It is believed to be in the Ising universality class and for the $d = 3$ dimensions, to which we restrict ourselves here, it has been used to extract the corresponding critical exponents to high accuracy, see e.g. Refs. (Hasenbusch *et al.*, 1999; Hasenbusch, 1999). Obviously, the HMC algorithm is not the most efficient choice for these computations. In the literature, local heat-bath and over-relaxation sweeps are combined with cluster updates.

The course resembled more a recitation class than a lecture, in the sense that after a short introduction the students were working on a problem sheet. The presentation here goes along the line of these problem sheets, essentially solving them. This might explain some repetitions and also the fact that many issues are not covered.

During the school, virtually all the code was written from scratch, with two exceptions: one is a routine which fills the neighbor field used to navigate the D dimensional lattice. The other is a routine that computes the action on a given field configuration. This essentially served the purpose of exemplifying the use of the hopping field. The starting code and also the final HMC are available under

`http://nic.desy.de/leshouches/`

or directly from the author.

The course proceeded as follows: in the first lesson, the model was defined and to gain some familiarity with the code, the measurement routines and the momentum refreshment were implemented. With the second sheet, the molecular dynamics was implemented and tested. The third class had the full HMC algorithm with the acceptance test, again checks of the code and first running studying thermalization. The fourth and final lesson was devoted to auto-correlation analysis and measurement of critical exponents. This text follows loosely this structure, keeping the style and also large parts of the text of the problem sheets. The actual tasks are given at the end of each section.

1.1 Definition of the model

We immediately define the model on a D dimensional L^D lattice with lattice spacing a and size $L = Na$. Thus the lattice points are given by

$$x = a(n_0, \ldots, n_{D-1}) \; ; \quad n_i \in \mathbb{N}_0 \; , \quad 0 \leq n_i < N \; .$$

Since this writeup is about numerical techniques, from now on a will be set to one. The real valued fields ϕ are living on the sites, $\phi_x \in \mathbb{R}$ and we adopt periodic boundary conditions: if $\hat{\mu}$ is the unit vector in μ direction, then $\phi_{x+L\hat{\mu}} = \phi_x$. The action S is given by

$$S(\phi) = \sum_x \left[-2\kappa \sum_{\mu=0}^{D-1} \phi_x \phi_{x+\hat{\mu}} + \phi_x^2 + \lambda(\phi_x^2 - 1)^2 \right] \; . \tag{1.1}$$

and expectation values can be computed using the path integral method

$$\langle A \rangle = \frac{1}{Z} \int \prod_x \mathrm{d}\phi_x \exp(-S(\phi)) A(\phi) \tag{1.2}$$

with the partition function $Z = \int \prod_x \mathrm{d}\phi_x \exp(-S(\phi))$.

The model has two well known limits: $\lambda = 0$ gives a Gaussian model, in the limit $\lambda \to \infty$ the Ising model is recovered. We will only do computations in three dimensions, where it has a second order critical line in the space spanned by the two coupling constants λ and κ.

1.1.1 Observables

We will be looking just at a few observables. The most important ones are powers of the magnetization m

$$m = \sum_x \phi_x \; . \tag{1.3}$$

Note that on a finite lattice $\langle m \rangle = 0$, because the action is invariant under $\phi_x \to -\phi_x$ and the fact that spontaneous symmetry breaking only occurs in an infinite volume. So interesting quantities are the magnetic susceptibility

$$\chi = \frac{1}{V} \langle m^2 \rangle$$

and the Binder cumulant U

$$U = \frac{\langle m^4 \rangle}{\left(\langle m^2 \rangle\right)^2} \ . \tag{1.4}$$

U is dimensionless and therefore can be used as a phenomenological coupling.

Other interesting quantities to look at are derivatives of observables with respect to κ. If A does not depend on κ, we can get them from correlations with the interaction term W

$$\frac{\partial}{\partial \kappa}\langle A \rangle = \langle WA \rangle - \langle W \rangle \langle A \rangle \qquad \text{with} \qquad W = 2 \sum_x \sum_{\mu=0}^{D-1} \phi_x \phi_{x+\hat{\mu}} \ . \tag{1.5}$$

1.2 Implementation

The starting point for the implementation are routines for the layout of the fields and the navigation of the D dimensional lattice, along with an example provided by a function to compute the action of a given field ϕ. They are available along with the code for the solutions under the URL given on page 1.

Most of it is in file `phi4.c`. It contains the main program, which at the moment does read in the basic parameters of the action, initializes the hopping field, fills the field ϕ with random numbers and computes the action S on a given field configuration ϕ.

1.2.1 Layout of the lattice

The ϕ field and the hopping field are global in `lattice.h`, where also the lattice size `L`, the dimension `D` and the volume `V` are defined. The lattice is ordered lexicographically, each point with coordinate $(n_0, n_1, \ldots, n_{D-1})$ gets assigned a unique index `j`. It can be computed by

$$j = \sum_{i=0}^{D-1} n_i L^i .$$

This way, the field ϕ_x of the Lagrangian is realized as the one dimensional field `phi[j]` with length $V = \prod_i L_i$ of the computer program.

Fortunately, the specific ordering of the points needs rarely to be understood. However, the hopping field `hop[V][2*D]` is important to understand: it is used to navigate the D dimensional lattice. The index of the neighbor of the point with index i in direction μ is `hop[i][mu]` in the forward direction $0 \le \mu < D$ and `hop[i][D+mu]` in the backward direction. If this point has coordinates (n_0, n_1, n_2), then the point with coordinates $(n_0, n_1 + 1, n_2)$ has index `jp=hop[i][1]`. Analogously, if we want to increase n_2 by one, then `jp=hop[i][2]`. The field also takes care of the periodic boundary conditions. If you want to go into the negative direction, the corresponding indices are in the upper D entries of the `hop` field. So $(n_0, n_1 - 1, n_2)$ has index `jp=hop[i][D+1]`.

As an example, the code for the computation of the action corresponding to a field `phi[]` is given in Figure 1.1. In line 11 we perform a loop over all sites and for each of them sum over the values of the field in the D positive directions. This corresponds

```
1 double action(void)
2 {
3    int i;
4    double J;
5
6    S=0;
7    for (i=0;i<V;i++) /* loop over all sites */
8    {
9        /*sum over neighbors in positive direction*/
10       J=0.0;
11       for (mu=0;mu<D;mu++) J+=phi[hop[i][mu]];
12
13       phi2=phi[i]*phi[i];
14       S+=-2*kappa*J*phi[i]+phi2+lambda*(phi2-1.0)*(phi2-1.0);
15   }
16   return S;
17 }
```

Fig. 1.1 Function for the compuation of the action.

to the μ sum in Eq. 1.1. From this quantity and the value of the field at the site, the action density can be computed and accumulated for the total value of the action.

The "tasks" given below ask for measurement routines to be implemented. They are simple modifications of the action routine with the magnetization just the sum over the field $\sum_x$ `phi[x]`. For the derivatives, the interaction term is needed, which is just the sum over `J*phi[x]` as in the subroutine for the action.

To get familiar with the code here are a few simple exercises

Exercise 1.1 Get the code, type `make phi4` to compile and then run it `./phi4 infile`. `infile` is the input file from which the parameters are read.

Exercise 1.2 Understand the `phi4.c` file. First `main()` then `action()` which computes the action S for the global field `phi[]`, in particular how the hopping field is used.

Exercise 1.3 Write a routine to measure the magnetization m defined in Eq. 1.3. Test this routine on field configurations ϕ of which you know the result.

Exercise 1.4 Write a routine to measure the other quantities necessary for the κ derivative of $\langle m^2 \rangle$ and $\langle m^4 \rangle$ using Eq. 1.5.

2
Hybrid Monte Carlo

Now that the fields are laid out and the action has been defined, we set out to implement the algorithm that generates a series of field configurations ϕ_i such that the expectation value in Eq. 1.2 can be estimated by a simple average over the measurements on those configurations

$$\langle A \rangle = \frac{1}{N} \sum_i A[\phi_i](1 + \mathcal{O}(1/\sqrt{N})) \ ,$$

where the error is purely statistical and is reduced with the square root of the number of measurements N. For this, an "exact" algorithm is needed, i.e. one which does not introduce a systematic error into this estimate, see Lüscher's lectures for more details.

HMC is an exact algorithm. It starts with enlarging the partition function by momenta π

$$Z = \int \prod_x \mathrm{d}\pi_x \prod_x \mathrm{d}\phi_x e^{-H(\pi,\phi)} \ ,$$

where for each site x, a momentum π_x is associated to the ϕ_x. The "Hamiltonian" H is given by

$$H(\pi, \phi) = \frac{1}{2} \sum_x \pi_x^2 + S(\phi) \ .$$

Obviously, expectation values of observables $A[\phi]$, which are functions of ϕ only, are unaltered. At first sight, the gain from this augmented partition function is unclear, however, a particular kind of field update becomes possible.

It is based on molecular dynamics (MD) evolution, which means that we treat this system in close analogy to classical mechanics, where the fields play the role of the (generalized) position and the potential energy is given by the action of the model. The system obeys Hamilton's equations of motion,

$$\frac{\mathrm{d}\phi}{\mathrm{d}\tau} = \frac{\partial H}{\partial \pi} \quad \text{and} \quad \frac{\mathrm{d}\pi}{\mathrm{d}\tau} = -\frac{\partial H}{\partial \phi} \ ,$$

where a "Monte Carlo" time τ is introduced. (To avoid any misunderstanding: these are not the equations of motion of the underlying theory, but those associated to the artificial Hamiltonian H.) From the solution of these equations of motion, a valid update algorithm can be derived, because from classical mechanics we know that they conserve the Hamiltonian H and the phase space volume, the latter by Liouville's theorem. Loosely speaking, a configuration of (π, ϕ) is equally likely to the (π', ϕ'),

which one gets by using (π, ϕ) as initial condition and solving the equations of motion for some time τ. In spirit of the analogy to classical systems, moving the fields in this way is called a trajectory of length τ. Since the energy H is conserved, during this evolution, this is a micro-canonical update.

But this is only one component of the HMC. The "hybrid" in the name of the algorithm comes from the fact that different algorithms are used for different parts of the partition function. For the momenta, heat-bath updates are employed. This is easy, because the π follow a simple Gaussian distribution independent of the ϕ. The fields ϕ are updated with (quasi) micro-canonical molecular dynamics evolution, solving the equations of motion. An algorithm which alternates between these two steps is the "Hybrid Molecular Dynamics" algorithm. The classic QCD citation is Ref. (Gottlieb *et al.*, 1987).

The HMD algorithm would be exact in case we could solve the equations of motion exactly. However, in general we have to use a numerical method to solve these equations and therefore integration errors occur. Given some properties of the integration procedure, namely the time-reversibility and the conservation of the phase space volume, this algorithm can be made exact with a final Metropolis step at the end of each trajectory. One considers the fields (π', ϕ') at the end of the trajectory as a proposal. The new field configuration is accepted with probability $\exp(-\Delta H)$, where ΔH is the difference of the Hamiltonian at the beginning and the end of the trajectory. If the proposal is rejected, we go back to the field ϕ and start again by the heat-bath for the momenta π. Note that in case the solution of the equations of motion is exact, we know that $\dot{H} = 0$, which implies ΔH is zero and the proposed change is always accepted.

Before starting the implementation, let us summarize the HMC algorithm for a single trajectory. This is then repeated N_{tr} times, where N_{tr} depends on the accuracy required and the amount of computer time allocated for the project.

1. Momentum heat-bath: Choose new random momenta according to the distribution $P(\pi_i) \propto \exp(-\pi_i^2/2)$.
2. Molecular dynamics evolution: Numerically solve the Hamiltonian equations of motion

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\tau}\phi_x(\tau) &= \frac{\partial}{\partial \pi_x} H(\pi(\tau), \phi(\tau)) \\ \frac{\mathrm{d}}{\mathrm{d}\tau}\pi_x(\tau) &= -\frac{\partial}{\partial \phi_x} H(\pi(\tau), \phi(\tau)) \end{aligned} \tag{2.1}$$

   for some interval of the fictitious time τ. This moves the fields from some initial (π, ϕ) to the proposed new fields (π', ϕ').
3. Acceptance step: Calculate the change in the Hamiltonian ΔH and accept the proposed new field ϕ' with probability

$$P_{\mathrm{acc}} = \min[1, \exp(-\Delta H)] \quad ; \quad \Delta H = H(\pi', \phi') - H(\pi, \phi) \ .$$

The implementation of these three steps will be discussed in the rest of the section, along with hints for the debugging of such code.

2.1 Momentum heatbath

The first step of the HMC algorithm is the momentum refreshment, i.e. the conjugate momenta π are filled with Gaussian random numbers. Random number generators by themselves are a complicated subject, which goes beyond the scope of this writeup. We therefore take as input a library routine which generates pseudo-random numbers, equally distributed in the range $[0, 1)$. A popular choice is `ranlux`, a high-quality random number generator(Lüscher, 1994) for which a C implementation is freely available.[1]

These have then to be transformed such that they follow a normal distribution, e.g. by the Box-Muller procedure: Given x_1 and x_2 which are drawn from a flat distribution $x_i \in [0, 1)$, then y_1 and y_2 with

$$y_1 = \sqrt{-2\ln(1-x_1)}\cos(2\pi(1-x_2))$$
$$y_2 = \sqrt{-2\ln(1-x_1)}\sin(2\pi(1-x_2))$$

are distributed according to $P(y_i) \propto \exp(-y_i^2/2)$. We always take $(1 - x_i)$ in order to exclude the zero in the argument of the logarithm.

Once such a routine has been implemented, it needs testing. The most obvious requirement is that the generated random numbers have the expected distribution. For this, it is convenient to histogram the output and compare with the expected form. To give such a comparison a statistical significance, the Kolmogorov-Smirnov test should be applied. From a practical point of view, this is sufficient in our context, *if* we assume that the underlying random number generator is of sufficiently high quality for our purposes.

Exercise 2.1 Write a routine which fills a vector of length n with double precision Gaussian random numbers distributed according to $P(x) \propto \exp(-x^2/2)$. Test this routine by filling many such vectors and histogramming. Does the distribution match your expectation? (Do this only if time allows, otherwise get the routine from me.)

Exercise 2.2 Now introduce the global momentum field `mom[V]` which will hold the momenta π conjugate to the field variables `phi[V]` and fill it with normally distributed random numbers. Write a routine which, given the `phi` and `mom` fields, computes the molecular dynamics Hamiltonian `H`.

2.2 Molecular dynamics

The second step in the HMC algorithm, the molecular dynamics evolution, is the part in which the fields ϕ are actually changed. In general, it is by far the most difficult to implement part of the whole algorithm. On the other hand, it is relatively easy to debug: it consists of the numerical solution of the classical MD equations of motion, for which we know that the Hamiltonian is conserved. By making the numerical integration more and more precise, we can therefore check the setup by observing an improvement in the energy conservation in accordance with the expected scaling from the integrator.

[1] `http://luscher.web.cern.ch/luscher/ranlux/index.html`

The approximate numerical solution of differential equations like Eqs. 2.1 is a vast field of current research, the coverage of which is beyond the scope of these exercises. For more information, see again Lüscher's lectures at this school and the book by Reich and Leimkuhler(Leimkuhler and Reich, 2005).

In many applications, the integration algorithm is constructed by a so-called splitting method, based on a decomposition of the Hamiltonian in exactly integrable pieces

$$H(\phi, \pi) = H_1(\pi) + H_2(\phi)$$

with $H_1(\pi) = \sum_x \pi_x^2/2$ and $H_2(\phi) = S(\phi)$. Also the splitting into more terms is possible and exploited in modern QCD codes. The algorithm consists of repeated application of the two elementary steps

$$\mathcal{I}_1(\epsilon) : (\pi, \phi) \to (\pi, \phi + \epsilon \nabla_\pi H_1(\pi)) \tag{2.2}$$

$$\mathcal{I}_2(\epsilon) : (\pi, \phi) \to (\pi - \epsilon \nabla_\phi S(\phi), \phi) \tag{2.3}$$

It turns out, that basically any combination of steps $\mathcal{I}_1(x)$ and $\mathcal{I}_2(x)$ leads to a legal integrator as long as the time shifts x add up to τ during a trajectory. A simple algorithm is the Störmer-Verlet method, in the QCD literature mostly referred to as the leap-frog, which corresponds to the application of

$$\mathcal{J}_\epsilon(\tau) = [\mathcal{I}_1(\epsilon/2)\mathcal{I}_2(\epsilon)\mathcal{I}_1(\epsilon/2)]^{N_s} \tag{2.4}$$

with $\tau = N_s\epsilon$ the length of the trajectory. Because it is symmetric under time-reversal, a potential deviation from the exact solution $\mathcal{O}(\epsilon)$ drops out and the leading violation due to the finite step-size ϵ is $\mathcal{O}(\epsilon^2)$. Observing this scaling behavior with varying step-size serves as the check of the correctness of the code discussed in the beginning.

The leap-frog integrator is still the main work horse of molecular dynamics simulations, however, in recent years in has been realized, that modified schemes can lead to significant improvement at little extra cost. One example is the so-called Omelyan integrator (Omelyan *et al.*, 2003) which reduces the coefficient of the ϵ^2 term and for one particular criterion an optimal scheme is given by

$$[\mathcal{I}_1(\xi\epsilon)\mathcal{I}_2(\epsilon/2)\mathcal{I}_1((1-2\xi)\epsilon)\mathcal{I}_2(\epsilon/2)\mathcal{I}_1(\xi\epsilon)]^{N_s}$$

with a tunable parameter ξ. The canonical value is $\xi \approx 0.1931833$.

Both integration methods, leap-frog and Omelyan's prescription, satisfy the requirements, which are needed for the final HMC to be exact: it is time reversible and the phase space measure is conserved, both properties of the exact solution. Reversibility means that if we perform one trajectory $(\pi^0, \phi^0) \to (\pi^1, \phi^1)$, then flip the momentum $\pi^1 \to -\pi^1$ and now use the same algorithm to run the trajectory back, we end up in exactly the same spot $(\pi^2, \phi^2) = (\pi^0, \phi^0)$, where we started. This is true at least in exact algebra. On a real computer with fixed precision arithmetic, this is spoiled by rounding errors, which come from the violation of associativity of the summation. Though the effect of this violation is in general tiny, the influence on the final measurements is unclear, but in most cases negligible. (In QCD simulations, however, the issue is serious, because chronological inversion techniques cause coupling from previous MC time steps.) Reversibility is a standard check of the code.

The second important property is that the map in phase space, which is defined by the integration algorithm like the one given in Eq. 2.4,

$$\mathcal{J}_\epsilon(\tau) : (\pi, \phi) \to (\pi', \phi')$$

conserves the phase space measure. As a consequence of this, the exponential of the change in the Hamiltonian $\Delta H = H(\pi', \phi') - H(\pi, \phi)$ is one on average

$$\langle \exp(-\Delta H) \rangle = \frac{1}{Z} \int [\mathrm{d}\pi][\mathrm{d}\phi] e^{-H(\pi,\phi)} e^{-\Delta H(\pi,\phi)} = \frac{1}{Z} \int [\mathrm{d}\pi'][\mathrm{d}\phi'] e^{-H(\pi',\phi')} = 1 \ ,$$

at least if the algorithm is exact, which is the case for the full HMC. This observable is therefore also routinely checked in simulations to provide yet another check for the code and the choice of parameters.

Here, a remark on the language used in the field is in place. In line with the language of classical mechanics, $F = -\nabla_\phi S(\phi)$ in Eq. 2.3 is called the "force". With a suitable norm, one can then speak of the size of such forces. Large forces typically require smaller step-sizes for the integration and therefore measuring them serves as a common guide to tuning the various parameters of modern HMC simulations.

2.2.1 Implementation

For the ϕ^4 Hamiltonian, the derivatives are easily computed. The derivative of the Hamiltonian with respect to the momentum π_x in Eq. 2.3 evaluates independently of the model to

$$\nabla_{\pi_x} H_1(\pi) = \pi_x \ . \tag{2.5}$$

The corresponding code can be found in Fig. 2.1. The expression for the force in Eq. 2.2, however, will depend on the particular choice of the discrete action. For the one given in Eq. 1.1, the forces read

$$\begin{aligned} \nabla_{\phi_x} S(\phi) &= -2\kappa \sum_{\mu=0}^{D-1} (\phi_{x+\hat{\mu}} + \phi_{x-\hat{\mu}}) + 2\phi_x + 4\lambda(\phi_x^2 - 1)\phi_x \\ &= -2\kappa J_x + 2\phi_x + 4\lambda(\phi_x^2 - 1)\phi_x \end{aligned} \tag{2.6}$$

Using $J_x = \sum_{\mu=0}^{D-1} (\phi_{x+\hat{\mu}} + \phi_{x-\hat{\mu}})$, the sum of the field ϕ on all neighboring sites. For this purpose, the hopping field `hop`, introduced in Sec. 1.2.1, also contains the indices of the sites in negative direction. For the site `x`, the sum can thus easily be computed by adding up `phi[y]` over all sites with indices `y=hop[x][mu]`, with `mu` running from 0 to $(2D-1)$. This is implemented in the code reproduced in Fig. 2.1.

Once this code is written, the routines have to be verified. Given a correct implementation of the computation of the action, a very stringent test is the behavior of the violation of energy along a trajectory. As already discussed, the change in $H(\pi, \phi)$ during the trajectory is expected to scale with ϵ^2, the square of the step-size of the integration algorithm. The result of the test at the parameters suggested below is displayed in Fig. 2.2. We observe exactly the expected behavior. The Omelyan integrator performs roughly a factor of two better than the leap-frog, even after normalization by

```
1  void move_phi(double eps)
2  {
3    int i;
4    for (i=0;i<V;i++) phi[i]+=mom[i]*eps;
5  }
6
7
8  void move_mom(double eps)
9  {
10   int i,mu;
11   double J, force;
12
13   for (i=0;i<V;i++)
14   {
15      J=0;
16      for (mu=0;mu<2*D;mu++) J+=phi[hop[i][mu]];
17
18      force=2*kappa*J-2*phi[i]-lambda*4*(phi[i]*phi[i]-1)*phi[i];
19      mom[i]+=force*eps;
20   }
21 }
```

Fig. 2.1 The two elementary updates corresponding to Eq. 2.2 (top) and Eq. 2.3(bottom) for the action given in Eq. 1.1, implementing Eq. 2.5 and Eq. 2.6 respectively.

the number of force evaluations (typically the most expensive part of the simulation) per step. However, one should stress that the fact that $\Delta H \propto \epsilon^2$ tests only whether force and action match, matching errors in both routines will remain undetected.

Exercise 2.3 Write routines which perform the two elementary updates in Eq. 2.3 and Eq. 2.2.

Exercise 2.4 Write a routine which repeats the sequence of these three steps N_s times. This moves the fields to time $\tau = N_s\epsilon$.

Exercise 2.5 Test this routine by measuring the Hamiltonian after each application of the three steps. Since they are supposed to solve the Hamiltonian equations, the energy H should be conserved up to $\mathcal{O}(\epsilon^2)$. Use trajectories of length 1 and test for various ϵ that this is indeed the case. Suggestion: use a 4^4 lattice, $\kappa = 0.18169$ and $\lambda = 1.3282$, starting from a random ϕ field. Do 1000 trajectories, measure the energy violation $|\Delta H|$ after each.

Exercise 2.6 An important property of the integrator is that it is reversible. So if we do a trajectory $(\pi, \phi) \to (\pi', \phi')$ and flip the momentum $\pi' \to -\pi'$, then the trajectory with $(-\pi', \phi')$ as initial values should lead to (π, ϕ). Check that this is the case. What spoils this?

Exercise 2.7 If you are ambitious: try the Omelyan integrator. Compare to the leap-frog.

2.3 Hybrid Monte Carlo

So far we have implemented the Hybrid Molecular Dynamics algorithm, which is inexact due to the integration errors of $\mathcal{O}(\epsilon^2)$. In a seminal paper on the Hybrid Monte

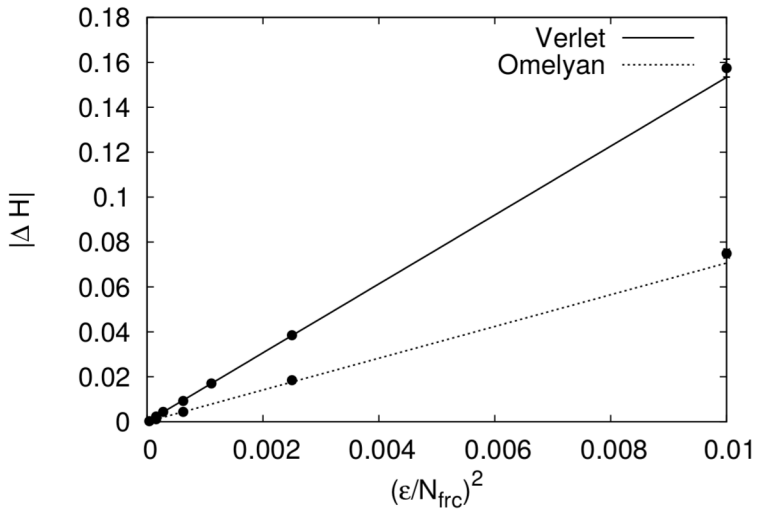

Fig. 2.2 Energy violation as a function of step size showing the expected quadratic convergence. Comparison between Verlet and Omelyan integrator.

Carlo algorithm (Duane *et al.*, 1987), Duane, Kennedy, Pendleton and Roweth realized that it can easily be made exact by a Metropolis acceptance step. Basically one measures the value of the Hamiltonian $H_i = H(\pi, \phi)$ at the beginning of the trajectory and saves the field ϕ. Then one performs the molecular dynamics evolution and again measures the Hamiltonian $H_f = H(\pi', \phi')$. This "proposed" new ϕ' is accepted with probability $\min[1, \exp(-\Delta H)]$ where $\Delta H = H_f - H_i$, else the new field is set to the initial field ϕ.

In a practical implementation, this acceptance step is realized by the following scheme: If $\Delta H < 0$, accept, else throw a random number r and accept if $\exp(-\Delta H) > r$, else reject the configuration. In case of rejection, copy back the ϕ at the beginning of the trajectory to the `phi[]` array.

More from the point of view of the implementation, a trajectory of the HMC algorithm therefore can be summarized in the following way

1. Momentum heat-bath on field `mom[]`.
2. Keep a copy of the current field `phi[]` in `phiold[]`.
3. Measure the current value of the Hamiltonian H_i.
4. Molecular dynamics with initial values `mom[]` and `phi[]`. These fields get overwritten by the values after the evolution.
5. Measure the value of the Hamiltonian H_f.
6. If the Metropolis test rejects the new configuration, copy `phiold[]` to `phi[]`.

This concludes the programming effort to implement the HMC algorithm for the ϕ^4 theory. An example for the implementation can be obtained at the URL given on page 1. Using this code, we can now simulate the model and measure expectation

values of the physics observables already implemented. The measurement would be step number seven in the list. It has to take place after the Metropolis step and after a potentially rejected configuration has been replaced. Typically, one does not measure after each trajectory. As will be discussed below, the configurations which are a few trajectories apart are correlated and the new measurement does not give much new information. The measurement frequency is therefore a balance between the cost of one measurement, the cost of a trajectory, the correlations between successive configurations and the amount of data to be stored. In our case, measurements are very cheap and since we are interested in the correlations, we can afford to measure after each trajectory. In QCD, measurements frequently cost a significant amount of computer time and consequently it is worth separating them by a couple of trajectories.

Exercise 2.8 Extend your program to the HMC algorithm: measure H and save the `phi[]` field at the beginning; perform the Metropolis step at the end.

Exercise 2.9 We also want to do physics measurements, so measure the magnetization m and the action S after each trajectory.

Exercise 2.10 Detour, if time allows: Computing the energy difference from H_i and H_f is susceptible to round-off errors. It is better to subtract first the energy densities and then perform the sum. Also doing the sums hierarchically might be advised, first summing over sub-lattices and then accumulating these results. Implement these improvements in your code.

2.4 Simulation

Given a correct algorithm, i.e. one which is ergodic, stable and without errors in the implementation, the Markov chain can be started from any configuration, or distribution of configurations. Applying the algorithm for a sufficiently large number of iterations will deplete the wrong contributions of this distribution and in the end, one is left with the correct distribution given by the theory one wants to simulate. This decay is exponential but the corresponding decay rates can be very small. The process is called "thermalization" and once it is over, the simulation is in "equilibrium".

In practical simulations, one measures several quantities in regular intervals. During the thermalization, their values show a systematic drift; once equilibrium is reached, they fluctuate around their "true" average. This can look rather differently for various quantities, but as long as there are these systematic movements in any quantity, equilibrium has not been reached.

If one is uncertain about equilibration, a way to proceed is to start from very different starting configurations and observe whether or not common values for the observables are reached.

Once in equilibrium, there is a further test of the correctness of the code: we know that $\langle \exp(-\Delta H) \rangle = 1$. Of course, as will be discussed in the last section, measurements are only meaningful with a well-determined error. For this we have to measure the auto-correlations in this particular observable using the methods discussed in that section. For a run on a lattice with $L = 6$, $\kappa = 0.185825$ and $\lambda = 1.1689$ with trajectories of length 1 using 10 steps of the leap-frog algorithm, we get a rate of

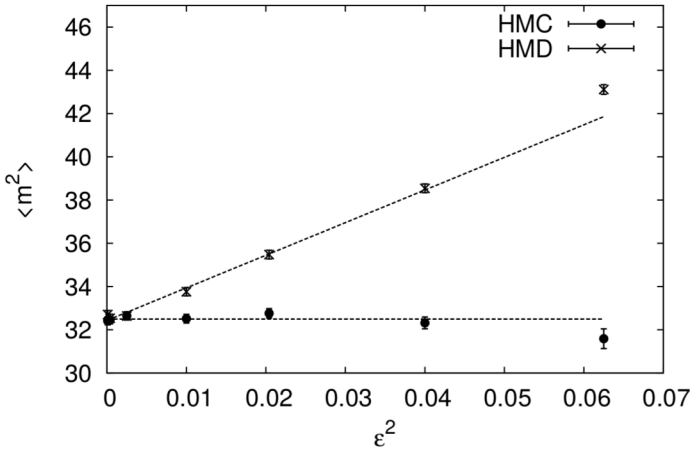

Fig. 2.3 Dependence of the magnetic susceptibility measured on the ensemble on the step size of the integration. The step-size error evident in the HMD algorithm is removed by the Metropolis step in the HMC. The parameters of the runs are given in the text.

acceptance of 87%. The exponential of the energy violation averages beautifully to 1, $\langle \exp(-\Delta H) \rangle = 1.0002(6)$, despite fluctuations of ΔH with $\langle \Delta H^2 \rangle = 0.1097(8)$.

Another demonstration that the acceptance step makes the HMC algorithm exact is to look at the dependence of observables with changing step size. In Fig. 2.3 we observe that the extracted value of $\langle m^2 \rangle$ is independent of it, at least within error-bars. The same algorithm without the Metropolis step, the Hybrid Molecular Dynamics algorithm, however, shows a clear deviation from this value which scales consistent with ϵ^{-2}.

In principle the HMC can be run at any step size, which leads to the question of its optimal choice. For most simulations, acceptance rates between 70% and 90% are good target values. Higher rates typically mean that too much effort is spent on the integration. Low values of the acceptance rate, however, frequently can be improved by relatively little additional effort. Also they might be an indicator for a problem in the numerical integration. The corresponding data for this simulation is plotted in Fig. 2.4. We show the acceptance rate as a function of the square of the step-size and see a linear deviation from one. Ultimately, what counts is the cost of producing an independent configuration, as will be explained in the next section. To first approximation this is the cost per accepted trajectory, which is shown in the right plot. We observe a relatively shallow minimum in the region between 50% and 90% acceptance.

2.5 Physics

Using this code, one is in the position to compute estimates for the observables one might be interested in. For this, of course, a method to compute their errors is needed, since averages without giving their uncertainty are of no use. This will be explained in the next section. The comparison of such expectation values to data from the literature

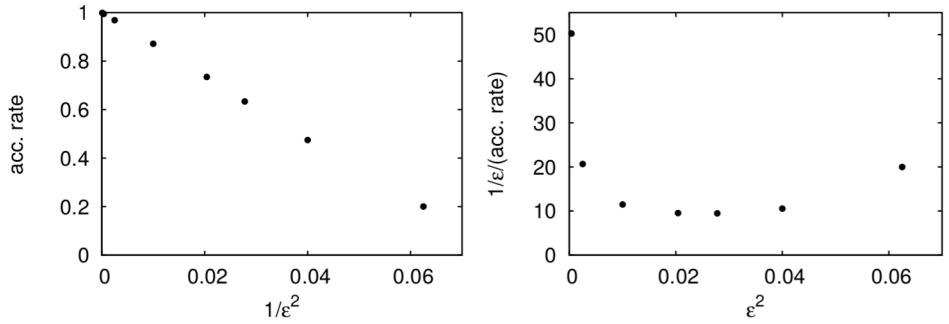

Fig. 2.4 Dependence of the acceptance rate on the step-size(left) and the cost per accepted trajectory also as a function of the step-size(right). The cost has a wide minimum which corresponds to acceptance rates between roughly 50% and 90%.

is as yet another test for the correctness of the code, and frequently a very important one. In our case, Refs. (Hasenbusch, 1999; Hasenbusch *et al.*, 1999) contain a large quantity of high precision data to serve this purpose.

In order to actually do some physics with the program, one might follow the lines of these references and study critical phenomena of the Ising universality class. In principle, the extraction of the critical exponents is possible, even though the algorithm is definitely not the fastest one for this particular model. Had this course been one session longer, the computation of some leading critical exponents could have been attempted. The reader is referred to the above mentioned publications for strategies. Fig. 2.5 has to serve as a demonstration that the physics comes out about right. What is shown is the average of $|m|$ as it crosses the phase transition. The absolute value is taken, because in finite volume $\langle m \rangle$ is zero. In infinite volume, the magnetization is zero above the critical temperature, and from the critical point on, it rises towards saturation. As we can see in the figure, this behavior is approached for L growing towards infinity.

Exercise 2.11 We do some initial test using $L = 6$, $\kappa = 0.185825$ and $\lambda = 1.1689$. This is pretty close to the critical line. Perform 5000 trajectories of length 1, $\epsilon = 0.05$, starting from a random initial configuration. Observe how the system thermalizes. Compare with runs at $\kappa = 0.1$ and $\kappa = 0.2$. How long (in MC time) do we have to wait until we can say that we are in equilibrium?

Exercise 2.12 Now make a longer run, eg. 10^5 trajectories. Start measuring after the thermalization has been completed. What are the values of the Binder cumulant, the magnetization and the action? To avoid problems with auto-correlations, combine the results from $N_{av} = 1000$ consecutive measurements and then do a naive error analysis on these values. What is the acceptance rate?

Exercise 2.13 Verify that $\langle e^{-\Delta H} \rangle = 1$.

Exercise 2.14 Convince yourself, that you actually have a correct algorithm. The results should not depend on the step-size. If the steps are too large, only the acceptance rates goes down, the average values of the observables should not be affected. Again, block 1000 measurements.

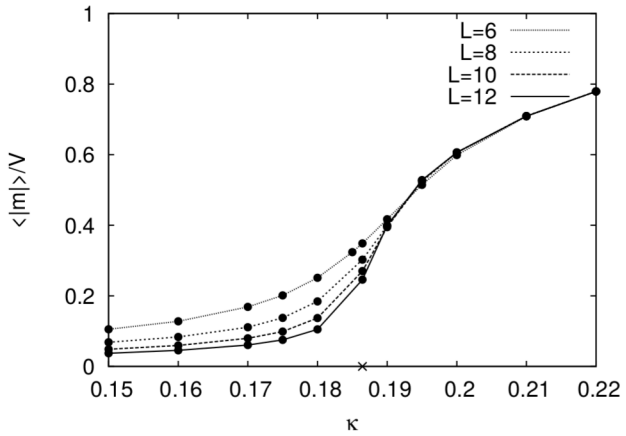

Fig. 2.5 Behavior of the magnetization $\langle m \rangle / V$ across the phase transition for different lattice sizes. In infinite volume, this should be zero for $\kappa < \kappa_{\mathrm{cr}}$ marked by the 'x' on the x-axis. This behavior is slowly approximated for growing L.

Exercise 2.15 Now change to $\lambda = 1.145$ where the critical point is according to Ref. (Hasenbusch, 1999) at $\kappa_c = 0.1864463(4)$. How does $\langle |m| \rangle / V$ change when you cross the phase transition. How does the picture depend on L?

3
Error Analysis

Most Monte Carlo algorithms produce a series of field configurations which are correlated among each other. These correlations die out with increasing Monte Carlo time separation, however, it still poses a problem to the estimation of the statistical errors of measured observables. Here, a brief summary of the method to deal with these correlations is given, for more detail see Refs. (Sokal, 1996; Wolff, 2004).

3.1 Theory

To be specific, our algorithm produces configurations $\phi^1 \to \phi^2 \to \cdots \to \phi^N$ on which observables A^α (labelled by an integer α) are measured. So we get a set of measurements $\{A_i^\alpha : i = 1, \ldots, N\}$. Let us assume that the thermalization process is already completed and the ϕ^i are in equilibrium. To quantify the correlations between the successive configurations one looks at the *auto-correlation function* $\Gamma_{\alpha\beta}$

$$\Gamma_{\alpha\beta}(t) = \langle (A_i^\alpha - \langle A^\alpha \rangle)(A_{i+t}^\beta - \langle A^\beta \rangle) \rangle \ . \tag{3.1}$$

Auto-correlations let this to be non-zero for $t > 0$. The brackets $\langle \cdots \rangle$ mean an average over repeated 'experiments', i.e. independent sets of N configurations. Of course, we almost never do this in real life, but use an estimator from one Markov chain

$$\bar{\Gamma}_{\alpha\beta}(t) = \frac{1}{N-t} \sum_{i=1}^{N-t} (A_i^\alpha - \bar{A}^\alpha)(A_{i+t}^\beta - \bar{A}^\beta) + \mathcal{O}(1/N) \tag{3.2}$$

with $\bar{A}^\alpha = \frac{1}{N} \sum_i A_i^\alpha$.

In most lattice simulations, we compute observables, which are functions of such averages of primary observables

$$F = F(A^1, \ldots, A^n)$$

from which we also need the derivatives $f_\alpha = \partial F / \partial A^\alpha$. For example, the Binder cumulant Eq. 1.4 is constructed from two primary observables $U = A_2/A_1^2$ with $A_1 = \langle m^2 \rangle$ and $A_2 = \langle m^4 \rangle$. Accordingly, the derivatives are $f_1 = -2A_2/A_1^3$ and $f_2 = 1/A_1^2$.

The obvious estimator of F is $\bar{F} = F(\bar{A}^1, \ldots, \bar{A}^n)$ and its error σ_F is given by(Wolff, 2004)

$$\sigma_F^2 = \frac{2\tau_{\text{int}}}{N} v_F \tag{3.3}$$

with the variance v_F given by

$$v_F = \sum_{\alpha\beta} f_\alpha f_\beta \Gamma_{\alpha\beta}(0)$$

and the integrated auto-correlation time for the observable F

$$\begin{aligned} \tau_{\mathrm{int},F} &= \frac{1}{2} + \frac{1}{v_F} \sum_{t=1}^{\infty} \sum_{\alpha\beta} f_\alpha f_\beta \Gamma_{\alpha\beta}(t) \\ &\equiv \frac{1}{2} + \sum_{t=1}^{\infty} \rho_F(t) \ . \end{aligned} \tag{3.4}$$

Since Eq. 3.3 is exactly the standard formula for the one sigma error, but where the number of measurements N is divided by $2\tau_{\mathrm{int}}$, one colloquially speaks of twice τ_{int} as the time to produce an independent configuration. It has to be stressed, however, that this statement can strongly depend on the particular observable F. The error of the normalized auto-correlation function ρ_F can also be given

$$[\delta\rho_F(t)]^2 = \frac{1}{N} \sum_{k=1}^{\infty} \left(\rho_F(k+t) + \rho_F(k-t) - 2\rho_F(t)\rho_F(k)\right)^2 \ .$$

Note that $\rho_F(t) = \rho_F(-t)$.

In a practical implementation, it is frequently tedious to compute first $\Gamma_{\alpha\beta}(t)$ via Eq. 3.1 and then the matrix element $f_\alpha \Gamma_{\alpha\beta}(t) f_\beta$ according to Eq. 3.4. The easier path is to first get $B_i = \sum_\alpha f_\alpha (A_i^\alpha - \bar{A}^\alpha)$ and from this immediately the relevant correlation function $\Gamma_F(t)$

$$\Gamma_F(t) = \frac{1}{N-t} \sum_{i=1}^{N-t} (B_i B_{i+t}) + \mathcal{O}(1/N) \ .$$

Because $\Gamma_{\alpha\beta}(t)$ is only poorly determined for large t, the sum for τ_{int} has to be truncated at some finite "window" W. Otherwise one would essentially sum up noise and increase the error of the computed $\tau_{\mathrm{int},F}$

$$\bar{\tau}_{\mathrm{int},F} = \frac{1}{2} + \sum_{k=1}^{W} \rho_F(t) \ . \tag{3.5}$$

By neglecting the contribution from $k > W$, one introduces a bias in the estimator. According to Madras and Sokal(Madras and Sokal, 1988), the variance of this estimator of τ_{int} is given by

$$(\delta\tau_{\mathrm{int},F})^2 \approx \frac{4W+2}{N} \tau_{\mathrm{int},F}^2 \ ,$$

which shows that without a finite W the variance would be infinite. The right choice of W amounts to a balance between bias and variance of the estimator of τ_{int}. Madras and Sokal (Madras and Sokal, 1988) use as W the first point where $W \geq c\tau_{\mathrm{int}}(W)$, e.g. with $c = 4$. For a single exponential decay, this leads to a 2% error on τ_{int}. Lüscher in Ref. (Lüscher, 2005) suggests to sum up to the smallest W such that $\sqrt{\langle\delta\rho(W)^2\rangle} \geq \rho(W)$, i.e. the point in ρ where the noise starts to overwhelm the signal.

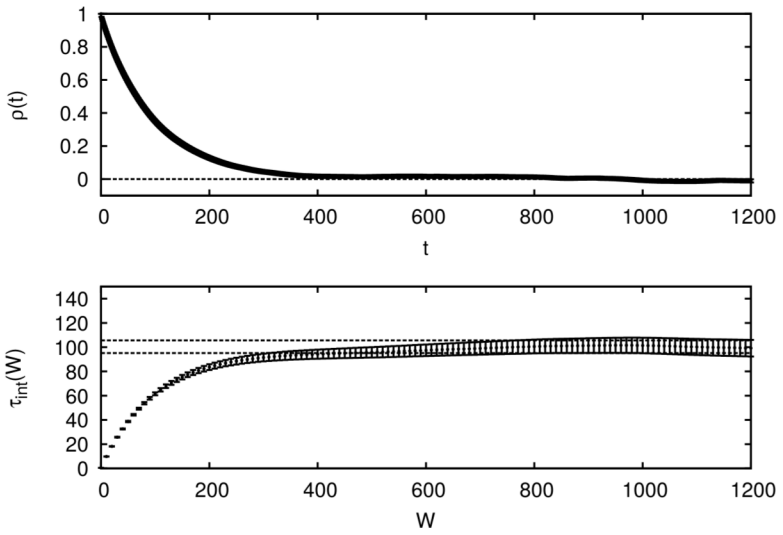

Fig. 3.1 Upper plot: Normalized auto-correlation function $\rho(t)$ of the magnetization m. The lower plot shows the accumulated sum for the computation of τ_{int} as function of the summation window W. The dashed lines indicate the 1σ range of its estimate.

3.2 Examples

An example of this procedure is given in Fig. 3.1. This is data from a run on a $L/a = 6$ lattice with $\lambda = 1.145$ and $\kappa = 0.18$ from one million trajectories. This is more statistics than many runs in QCD. One can observe a nice exponential fall-off in the normalized auto-correlation function $\rho(t)$. In the lower plot the partial sum of τ_{int} as in Eq. 3.5 up to a window W is done. This saturates at around $W = 800$, a point where the measured $\rho(t)$ is still measured well enough to be different from zero.

The measurement of auto-correlation times can also serve as a rational of the choice of the trajectory length τ. So far it has been a free parameter of the algorithm. The basic idea is that for very short trajectory lengths (maybe one short elementary leap-frog step only), the momentum are refreshed very frequently and one moves in a new, arbitrary direction. This means the system performs a random walk, which is known to be inefficient as the distance from the origin after N_s steps scales with $\tau\sqrt{N_s}$; the amount of work, however, goes with τN_s. Scaling very short τ by a factor of x should therefore result in an improvement by $\sqrt{x}$, keeping the cost constant. At some point, this argument breaks down because the random walk regime is left. Then longer trajectories just cost more without any further improvement in the auto-correlations. In Fig. 3.2, this behavior is shown for the auto-correlation time in units of molecular dynamics time, i.e. cost. One observes a clear minimum around $\tau = 2$. However, in a range between $\tau = 1$ and 4 remains within 25% of the optimum.

Auto-correlation times depend strongly on the physical system, the observable and

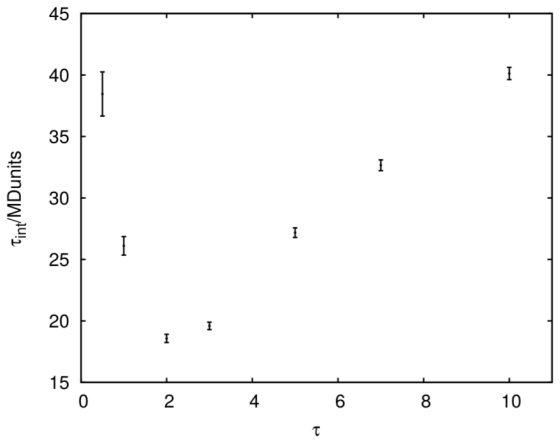

Fig. 3.2 Auto-correlation time of m^2 in units of Monte Carlo time as a function of trajectory length. In this simulation, trajectory length is almost directly proportional to the cost. The optimal trajectory length for this observable is therefore around $\tau = 2$.

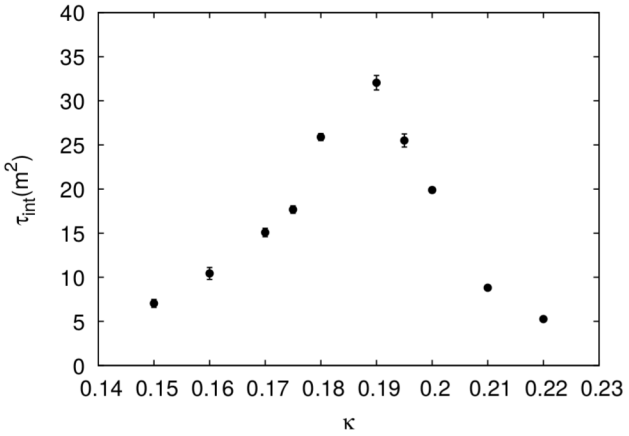

Fig. 3.3 Auto-correlation time of m^2 in units of Monte Carlo time as a function of κ for a $L = 6$ lattice and $\lambda = 1.145$. The critical point is at $\kappa_c = 0.1864463(4)$.

the algorithm. They are therefore extremely difficult to predict. What virtually all algorithms have in common, though, is the fact that towards the critical point, they perform worse. This phenomenon is called critical slowing down, i.e. the integrated auto-correlation times of the observable in question increases as the critical point is approached. An example of this is shown in Fig. 3.3 with the critical point $\kappa_{\mathrm{cr}} = 0.1864463(4)$ for $\lambda = 1.145$ taken from Ref. (Hasenbusch, 1999). In general, one expects a behavior similar to critical phenomena, i.e. a scaling with the correlation length according to a power law with a *dynamical* critical exponent z: $\tau_{\mathrm{int}} \propto \xi^z$.

To summarize, the determination of integrated auto-correlation times is a central part of the planning and analysis of all Monte Carlo simulations. Before even starting the runs, one needs to have an idea of the magnitude of the τ_{int} of the observables to be measured in order to decide whether the computation is feasible at all. Once the measurements are done, not all auto-correlation functions will look as in Fig. 3.1. The errors are typically larger and also long tails can make a precise determination of τ_{int} difficult. However, without the auto-correlation time, we cannot give an error and without an error, the measurement is without meaning.

Exercise 3.1 Write a program which reads in the MC time history of the observables A^i and computes the expectation value of a derived quantity $\bar{F}$ and its error.

Exercise 3.2 Take a sample equilibrium history of m and m^2 and compute the normalized auto-correlation functions ρ. Plot them and look at the exponential fall-off. Then compute $\tau_{\text{int}}(W)$ and do some experiments regarding the summation window W. Suggestion: Use $L/a = 6$, $\lambda = 1.145$ and $\kappa = 0.18$ with 10^6 measurements.

Exercise 3.3 Study the dependence of the auto-correlation time of m^2 on the trajectory length and the step size. What are issues in choosing it for production running?

Exercise 3.4 Now extend the program to analyze the Binder cumulant. Compute the necessary derivatives analytically and modify the `observable()` and `derivative()` routines.

Exercise 3.5 In the vicinity of phase transitions auto-correlation times tend to increase for most algorithms. This is called critical slowing down. Study τ_{int} of m and the Binder cumulant as κ crosses the critical point of $\kappa_c = 0.1864463(4)$ at $L/a = 6$, $\lambda = 1.145$.

Exercise 3.6 Try to get the exponential auto-correlation time τ_{exp}, for which should hold $\rho(t) = C \exp(-t/\tau_{\text{exp}})$ for $t \to \infty$. (To be more precise, it is the supremum over all possible observables.)

4
Summary

This course covers the implementation of an HMC for ϕ^4 theory, the measurement of simple observables and the data analysis. For the four sessions available, in which most of the programming was done, this was already a dense program. The next step could be the actual analysis of critical properties of this theory. However, one would like to use a better algorithm for this to be a satisfying experience.

Many other interesting topics are not covered as well. For example, one could have split the action and introduced multiple time scales for the integration, have gone more into the analysis of round-off errors or tried more advanced integrators. Also a comparison to single variable update algorithms would be interesting. However, I still hope that to those, who have never done a complete simulation, it gives an impression of how a simulation with an HMC algorithm is set up and how data is analyzed.

Probably my biggest regret is that these exercises are very conservative by focussing on HMC instead of broadening the knowledge of alternative methods. The strength of the algorithm is that it can be used to simulate any theory with continuous variables, also when many degrees of freedom are coupled; a situation where single variable updates typically fail. However, at least in the traditional form presented here, virtually no information about the theory is injected in the setup. This can be considered an advantage, because there is no risk of a bias in the result. Still, recent improvements of the HMC algorithms have shown that using knowledge about the physics of the system can lead to significant improvements of the performance. Hasenbusch's mass preconditioned HMC(Hasenbusch, 2001) or Lüscher's DD-HMC (Lüscher, 2005) are prime examples of this. Hopefully, the future will bring methods, which are even more adapted to the problem in question and are even more powerful for the simulation of QCD or other field theories.

Acknowledgements

I want to thank the organizers of this school for inviting me to give this course, Martin Hasenbusch for interesting discussions during its preparation and Rainer Sommer for useful remarks on the manuscript.

References

S. Duane, A.D. Kennedy, B.J. Pendleton and D. Roweth, "Hybrid Monte Carlo," Phys. Lett. B **195** (1987) 216.

S. A. Gottlieb, W. Liu, D. Toussaint, R. L. Renken and R. L. Sugar, "Hybrid Molecular Dynamics Algorithms for the Numerical Simulation of Phys. Rev. D **35** (1987) 2531.

M. Hasenbusch, "A Monte Carlo study of leading order scaling corrections of phi**4 theory on a three dimensional lattice," J. Phys. A **32** (1999) 4851 [arXiv:hep-lat/9902026].

M. Hasenbusch, K. Pinn and S. Vinti, "Critical exponents of the three-dimensional Ising universality class from finite-size scaling with standard and improved actions," Phys. Rev. B **59** (1999) 11471.

M. Hasenbusch, "Speeding up the hybrid Monte Carlo algorithm for dynamical fermions," Phys. Lett. **B519** (2001) 177-182. [hep-lat/0107019].

Leimkuhler, B. and Reich, S., Simulating Hamiltonian Dynamics. 2005. Cambridge University Press, Cambridge.

M. Lüscher, "A Portable high quality random number generator for lattice field theory simulations," Comput. Phys. Commun. **79** (1994) 100 [arXiv:hep-lat/9309020].

M. Lüscher, "Schwarz-preconditioned HMC algorithm for two-flavour lattice QCD," Comput. Phys. Commun. **165** (2005) 199 [arXiv:hep-lat/0409106].

M. Lüscher, "Computational Strategies in Lattice QCD," arXiv:1002.4232 [hep-lat].

N. Madras and A. D. Sokal, "The Pivot algorithm: a highly efficient Monte Carlo method for selfavoiding walk," J. Statist. Phys. **50** (1988) 109.

I.P. Omelyan, I.M. Mryglod and R. Folk, Comput. Phys. Commun. **151** (2003) 272.

A. D. Sokal, Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms, Lectures ath the Cargeèse Summer School, 1996

U. Wolff [ALPHA collaboration], "Monte Carlo errors with less errors," Comput. Phys. Commun. **156** (2004) 143 [Erratum-ibid. **176** (2007) 383].