Tutorial 6

Monte Carlo: The Ising model

Peter Košovan[*], Marcello Sega

February 4, 2010
ICP, University of Stuttgart

The purpose of this tutorial is twofold. In the first place, we will use the 2D Ising model to study influence of temperature on magnetization. Second, we will take the advantage of the fact that the Ising model has been solved analytically and therefore the results of our simulations are known beforehand. This will be useful when trying to understand how systematic and statistical errors of the simulation results depend on the length of simulation, system size and equilibration.

1 The model

Ising model is one of the heavily studied models in statistical physics, in part thanks to its relative simplicity. It consists of spins positioned on a lattice with nearest neighbour interactions, resulting in the following partition function:

$$Z_\mathrm{I} = \sum_{\{\sigma_i\}} \exp(-\beta H_\mathrm{I}), \qquad H_\mathrm{I} = -\sum_{\langle ij \rangle} \sigma_i \sigma_j, \qquad \sigma_i = \pm 1 \tag{1}$$

where σ_i are individual spin values, and $\beta = J/k_\mathrm{B}T$ is the inverse temperature in natural units. The angular brackets $\langle\cdot\rangle$ indicate that the summation is performed only over the nearest-neighbours on the lattice. The spins are placed on a cubic lattice with edge length L in D dimensions with the total volume of $V = L^D$. For $D = 2$ the Ising model has been solved exactly for both finite and infinite values of L. For $D = 3$ only approximate numerical solutions are currently available. For more details on the Ising model we refer the reader to the available literature. Here we pick up just a couple of points which we will need later on.

[*]kosovan@icp.uni-stuttgart.de

Among the observables one can compute for the Ising model, in our simulation we will follow internal energy per site, e, and magnetization, m. The internal energy per site is defined as

$$e = E/V, \qquad E = \langle H_\mathrm{I} \rangle \,, \tag{2}$$

and the magnetization

$$m = M/V = \langle |\mu| \rangle \,, \qquad \mu = \sum_i \sigma_i / V \,. \tag{3}$$

Magnetization in the Ising model on an infinite lattice is unity at $T = 0$, decreases with increasing temperature and vanishes in a first-order phase transition at critical temperature, T_c. At this temperature, correlation length, ξ diverges. This means that fluctuations on all length scales exist close to critical point and thermodynamic observables diverge following power-law dependence, for example of the type

$$\xi \sim t^{-\nu}, \qquad m \sim t^{-\beta_m}, \qquad t \equiv |1 - T/T_\mathrm{c}| \,. \tag{4}$$

For $2D$, the values of T_c and the critical exponents β_m and ν are known analytically as well as the critical temperature:

$$\beta_\mathrm{c} = \frac{J}{k_\mathrm{B} T_\mathrm{c}} = \frac{\log_e(1+\sqrt{2})}{2}, \tag{5}$$

Note that symbol β is for historical reasons used both for the inverse temperature and the critical exponent for magnetization. To clearly distinguish between them, in this worksheet, we will use β_m for the cricital exponent for magnetization.

Simulations can only be performed on finite lattices, usually with periodic boundary conditions. By construction, these cannot handle infinite-ranged correlations close to the critical point. Therefore, the results obtained from finite-lattice simulations will systematically differ from the infinite limit.

2 The simulation program

The code used for this tutorial is a simulation of Ising model on a $2D$ lattice with periodic boundary conditions. We use local moves which are updated according to the Metropolis algorithm. This is not the most efficient method but has the advantage of beaing easily understandable.

Task: (2 points)

Implementation of the Metropolis criterion

- *Read the code in file* `ising.c`*, and complete the function* `evaluate_mc_step()` *which implements the Metropolis update on one spin. This part is marked in the code for you.*

To compile the code, use the following command:

```
> gcc ising.c -o ising -lgsl -lgslcblas
```

Recall that according to the Metropolis acceptance rules, if the energy of the new configuration is lower than that of the old one, the new configuration is automatically accepted. When the energy of the new configuration is higher than that of the old one, it is accepted with the probability proportional to the Boltzmann factor:

$$\mathcal{P} = \exp\left(-\beta(E_{\text{new}} - E_{\text{old}})\right)$$

In practice it is realized so that we draw a random number between 0 and 1 from a uniform distribution and check if it is smaller than the Boltzmann factor.

To obtain help on input parameters, run the program without arguments, executing

```
> ./ising
```

You can start the simulation from a completely random configuration (hot) or completely aligned configuration (cold). Running the same simulation with different starting conditions should help you to tell if your system is equilibrated.

The program automatically calculates some averages from the whole simulations run, namely magnetization $\langle m \rangle$ and its square $\langle m^2 \rangle$ by taking into account *all* configurations, including equilibration. It also calculates an estimate of error of $\langle m \rangle$ based on the assumption that individual samples are uncorrelated. Magnetization and energy of each sample are saved in the file `magnetization.dat` and images are produced to visualize the results. You can produce a simple animation using the following command (be patient, it takes a couple of seconds, especially if you have many or big images)

```
> animate -delay 10 *.ppm
```

Make sure you remove old images from the working directory before running a different simulation! You may add an option `-geometry 200x200` to define any convenient size of the animation window on your screen (in pixels). The snapshots should be similar to what is shown in figure 1, which display uniformly magnetized sample at low temperature, domains in the proximity of the phase transition, and random magnetization at higher temperatures.

Task: (2 points)

Visual inspection of the results

- *Perform the simulation at temperatures $T = 0.6,\ 2.4,\ 3.5$ with the following parameters:* `n_sweeps=1000`, `lattice_size=500`, `n_images=200`.
- *Explain what you observe in the animations in relation to formation of magnetic domains.*
- *Plot magnetization as a function of number of sweeps for all three temperatures. Using the plots, estimate how many sweeps it takes to equilibrate the system. If it is not equilibrated within the runtime, just say it. In this task you are not expected to run longer simulations to equilibrate the system.*

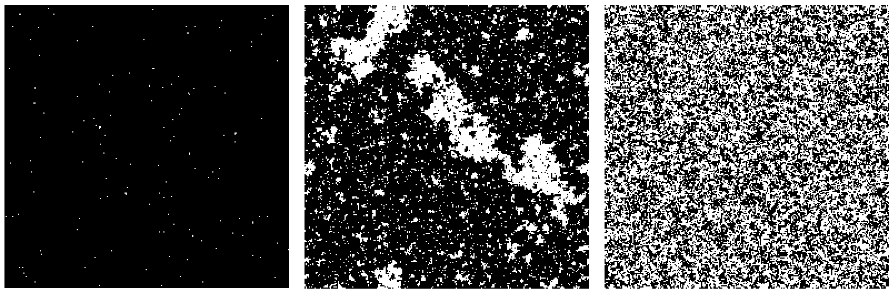

Figure 1: Configuration snapshots at three different temperatures. From left to right: $T \ll T_\mathrm{c}$, $T \approx T_\mathrm{c}$, $T \gg T_\mathrm{c}$.

3 Analysis of results

In any scientific measurement, two types of errors exist: systematic and random. Systematic errors arise due to a bias or imperfection in the measurement. As a consequence, the measured value is shifted with respect to the true value in one specific direction. For example, when we are simulating a finite system with periodic boundary conditions to mimick an infinite system, we are introducing a systematic error by neglecting correlations larger than the box size. On the other hand, random errors originate from various random disturbances and do not have a preferred direction. Unlike systematic errors, they tend to cancel out when performing more measurements. Systematic errors can often (but not always) be avoided, but random errors cannot. If the error cannot be avoided, it is necessary to have control over its magnitude. In the following, we will analyze systematic error due to finite lattice size and random errors.

For those who are more interested in error analysis, we recommend reading the paper by Janke [1] where it is described in more detail. The following paragraphs are largely based on this text, but only discuss points necessary for our case. Estimate $\bar{\mathcal{O}}$ of the expectation value $\langle \mathcal{O} \rangle$ of the observable $\mathcal{O}$ is computed as

$$\bar{\mathcal{O}} = \frac{1}{N} \sum_{j=1}^{N} \mathcal{O}_j \,. \tag{6}$$

Besides the average value itself, we are always intereted also in its statistical error,

$$\sigma^2_{\bar{\mathcal{O}}} = \left\langle \left[\bar{\mathcal{O}} - \langle \bar{\mathcal{O}} \rangle \right]^2 \right\rangle \tag{7}$$

For N uncorrelated samples, $\mathcal{O}_j$, it is simply

$$\sigma^2_{\bar{\mathcal{O}}} = \sigma^2_{\mathcal{O}_j} / N \tag{8}$$

where $\sigma^2_{\mathcal{O}_j} = \left\langle \mathcal{O}_j^2 \right\rangle - \langle \mathcal{O}_j \rangle^2$ is the variance of individual measurements. Since in our simulations new configurations are based on the previous ones, there is always a certain level of correlation between successive measurements. If we should use the error estimate based on Equation 8, we would underestimate the true statistical error. In principle we could increase the sampling interval so that our samples are *almost* uncorrelated, but this would be inefficient. Instead, it is more practical to try and estimate the extent of correlation and to correct the error estimates.

3.1 The binning method

In the binning method, we divide our data into N_B non-overlapping blocks of size k, such that $N = kN_B$. Then we define per-block averages

$$\mathcal{O}_{B,n} = \frac{1}{k}\sum_{i=1}^{k} \mathcal{O}_{(n-1)k+i}\,, \qquad i = 1, \ldots, N_B \tag{9}$$

If the block size is much larger than autocorrelation time, $k \gg \tau$, we can assume the blocks to be uncorrelated and use the formula from Equation 8 to estimate their variance:

$$\sigma^2_{\bar{\mathcal{O}}} = \sigma^2_B / N_B = \frac{1}{N_B(N_B-1)} \sum_{n=1}^{N_B} (\mathcal{O}_{B,n} - \bar{\mathcal{O}}_B)^2\,. \tag{10}$$

The estimate of the integrated autocorrelation time can be then obtained from relation

$$2\tau_{\mathcal{O},\mathrm{int}} = k\sigma^2_B / \sigma^2_{\mathcal{O}_i}\,. \tag{11}$$

This tells us, after how many samples the correlations are lost. The effective number of uncorrelated samples can be then computed as $N_{\mathrm{eff}} = \texttt{n\_sweeps}/\tau_{\mathcal{O},\mathrm{int}}$. Note that correlation times for different observables in the same simulation are usually different! As a rule of thumb, one needs about 10^3 uncorrelated samples for relative accuracy of about 5–10%.

3.2 The Jackknife method

A similar method is called the Jackknife, which takes larger blocks rather than small ones. This reduces the bias when non-linear combinations of basic observables are considered. Similar to the previous case, we take N_B Jackknife blocks, defined as

$$\mathcal{O}_{J,n} = \frac{N\bar{\mathcal{O}} - k\mathcal{O}_{B,n}}{N-k}\,, \qquad n = 1, \ldots, N_B\,. \tag{12}$$

And the estimator of variance is then

$$\sigma^2_{\bar{\mathcal{O}}} = \sigma^2_B / N_B = \frac{(N_B-1)}{N_B} \sum_{n=1}^{N_B} (\mathcal{O}_{J,n} - \bar{\mathcal{O}}_J)^2\,. \tag{13}$$

3.3 The analysis program

The evolution of magnetization and interaction energy per site are stored in the file `magnetization.dat`. Together with the simulation program, you are provided with an analysis tool, which you can compole using

```
  > gcc ./analyze.c -o analyze -lm
```

It is made to analyze a time-series of observables, excluding a given number of samples from the beginning (equilibration) and compute averages and error estimates from the rest. Two methods of error analysis are implemented in the provided code. The first one estimates the error according to Equation 8, the second is the binning method (Equation 10).

Task: (2 points)
Implementing the Jackknife method.

- *In the code of* `analyze.c` *implement the function* `analyze_jack`*. Hint: you can use the code of the function* `analyze_bins` *as a template.*

Task: (4 points)
Observables and error analysis. Once you have a working code, you may play around and test some of the properties of the error estimates.

- *Perform simulations at* $T = 2.35$ *for lattice sizes* $L = 10,\ 20,\ 40,\ 80,\ 160$. *Make sure that your system is equilibrated, by checking that you reach the same final state starting both with the* hot *and* cold *initial configuration.*

- *Analyze the results. For each system, provide the following parameters in a table: lattice size, total duration, duration of equilibration, number of blocks. For each observable (*m *and* e*) provide its average, estimated error and correlation time using both the binning and Jackknife method. To ensure reliability of the error estimation, use parameters such that* $N_B > 10$ *and* $k > 10\tau_{\mathcal{O},\mathrm{int}}$. *Make each simulation long enough so that the estimated error is less than 5%.*

- *Plot the average value of* m *and* e *(with error bars) as a function of* L. *Discuss how systematic error changes with* L. Hint: for scaling plots, use double-logarithmic scale. Then all power-laws show up as straight lines.

- *Both the blocking and the Jackknife method are relatively robust. The estimated error and correlation time only weakly depends of the number of blocks, as long as* $k \gg \tau$ *and* $N_B \gg 1$. *Show that by plotting* $\tau_{\mathcal{O},\mathrm{int}}$ *as a function of* N_B *for the largest lattice. Discuss what happens when* $N_B \approx N$ *and* $N_B \approx 1$. Hint: choose values of N_B which are equidistant on the logarithmic scale, e.g. 1, 2, 4, 8, 16, ...

Optional task: (2 bonus points)

Autocorrelation time and lattice size

- *Also autocorrelation time follows a power-law relation with the lattice size, $\tau \sim L^z$. Try to estimate the value of critical exponent z from your simulations in a similar fashion as β_m was estimated in Homework 4. Plot $\tau(L)$ for three different temperatures: $T = 1.5, 2.35, 2.6$.*
- *By performing estimation of τ for several different values of N_B, show that your estimate does not depend on your choice of N_B. Provide the corresponding plot.*
- *How long would it take to simulate a system with $L = 10^4$ at these temperatures with our current program, if we require the same relative accuracy?*

Optional task: (2 bonus points)

Master curve for finite size scaling.

- *When plotting $mL^{\beta_m/\nu}$ against $tL^{-\nu}$, for different values of L and t, all data fall on a single master curve. Use all your data from Homework 4 and perform several additional simulations at other values of t and L. Knowing that $\nu = 1$, try to estimate the value of β_m in the following way: make plots of mL^a against $tL^{-\nu}$ with different values of a and select one where all data points are on the same curve. Always plot the data with errorbars, so that one can visually check if they fit within the estimated error. Provide the best-looking plot and your estimated value of β_m.*

References

[1] `http://www.fz-juelich.de/nic-series/volume10/janke2.pdf`