Christian Schäfer: cschaefer@th.physik.uni-frankfut.de

Phase transition of the 2d Ising Model via Monte Carlo simulations

1 Introduction

In this project we compute the critical temperature for the two dimensional Ising Model[1] phase transition using Monte Carlo simulations. Let $s_{i,j}$ denote a spin state at lattice coordinates i and j having either spin up or spin down, $s_{i,j} = \pm 1$. The Hamiltonian or total energy of the system in a particular state $\{s_{i,j}\}$ is

$$H(\{s_{i,j}\}) = -J \sum_{i,j} s_{i,j} \left(s_{i+1,j} + s_{i-1,j} + s_{i,j+1} + s_{i,j-1} \right), \tag{1}$$

assuming periodic boundary conditions and only nearest neighbour interactions with a coupling J. The probability of finding the system in any particular state $\{s_{i,j}\}$ is given by

$$W(\{s_{i,j}\}) = \frac{1}{Z(\beta)} \exp\left[-\beta H(\{s_{i,j}\})\right], \tag{2}$$

where $\beta = 1/(k_{\rm B}T)$ with T the temperature, $k_{\rm B}$ Boltzmann's constant and

$$Z(\beta) = \sum_{\{s_{i,j}\}} \exp\left[-\beta H(\{s_{i,j}\})\right], \tag{3}$$

the partion function.

The computation of macroscopic quantities like the magnetization requires an integration/sum over all spin configurations weighted with their respective probability. The Metropolis-Hastings[2] algorithm allows for an effective computation of these multidimensional integrals/sums by sampling their probability distributions by Markov chains[3]. Thus the mean magnetization can be computed from

$$\langle M \rangle = \frac{1}{N} \sum_{\{s_{i,j}\}} M(\{s_{i,j}\}), \tag{4}$$

where we average over N configurations $\{s_{i,j}\}$ generated according to the probability $W(\{s_{i,j}\})$.

2 Implementation

The following steps guide you through the development ouf your own Monte Carlo simulation for the 2d Ising Model.

2.1 Pseudo random number generator

Add a pseudo random number generator to your code, which creates pseudo random numbers r uniformly distributed between $r \in [0, 1)$:

double giveRandomNumber ().

Regarding our problem the ordinary generator[4] is sufficient.

[1] http://en.wikipedia.org/wiki/Ising_model
[2] http://en.wikipedia.org/wiki/Metropolis-Hastings_algorithm
[3] http://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo
[4] http://www.cplusplus.com/reference/cstdlib/rand/

2.2 Geometry of the lattice

Create a one-dimensional array, which contains the spin alignment for every site of your $L \times L$ lattice. To access this array you want to introduce a superindex SI mapping your site coordinates to, $SI(i, j) : \mathbb{Z} \times \mathbb{Z} \to \mathbb{N}$. Provide a method of the form

int getSuperIndex (int i, int j).

Note, this method is a suitable place to include periodic boundary conditions, i.e. that $i + n \times L = i$ and $j + n \times L = j$ for $n \in \mathbb{Z}$.

2.3 Cold start and magnetization

Initialize your spin configuration by a cold start, i.e. setting all spins to one $s_{i,j} = 1 \; \forall \; i, j$:

void coldStart (int latticeSpin []).

Implement the magnetization $M = \frac{1}{L^2} \sum_i s_i$ as an observable:

double computeMagnetization (int latticeSpin[]).

2.4 Metropolis-Hastings algorithm

Update your lattice of spins according to the Metropolis-Hastings algorithm:

void updateLattice (int latticeSpin[], double effectiveBeta).

The idea of the Metropolis-Hastings algorithm is to replace one spin state of your lattice by $s_{i,j} \to s'_{i,j} = -s_{i,j}$ and accept this step with a probability

$$W(s_{i,j} \to s'_{i,j}) = \begin{cases} 1 & \text{if } \delta H < 0 \\ \exp\left(-\beta \delta H\right) & \text{else} \end{cases}, \tag{5}$$

where $\delta H = H(s'_{i,j}) - H(s_{i,j})$ corresponds to the change in energy. Regarding perfomance of your code simplify δH analytically before implementing. Use your pseudo random number generator to implement the acceptance step.

Then produce a number of N configurations by repeating this update step for every lattice site and N times for the whole lattice. Plot the magnetization M for every update steps to observe how the system develops. When should you start estimating observables, e.g. the mean magnetization $\langle M \rangle$?

3 Determination of the critical temperature

3.1 Critical temperature of the phase transition

Introduce an effective temperature β_{eff}, which includes the coupling J and the inverse temperature β, i.e. $\beta_{\text{eff}} = J\beta$. Then compute the absolute mean magnetization $| \langle M \rangle |$ based on the configurations created in your Metropolis-Hastings algorithm and plot it against different effective temperatures β_{eff}. Do you see the phase transition (c.f. fig. 1)? Improve your result by performing 10 lattice updates between each computation of the mean magnetization.

3.2 Magnetic susceptibility

A more sophisticated way to compute the critical temperature is given by determining the magnetic susceptibility $\chi_M = \langle M^2 \rangle - \langle M \rangle^2$, which diverges at the phase transition. Add the computation of this observable to your code and plot its value for different effectives temperatures β. Estimate the critical temperature β_c.

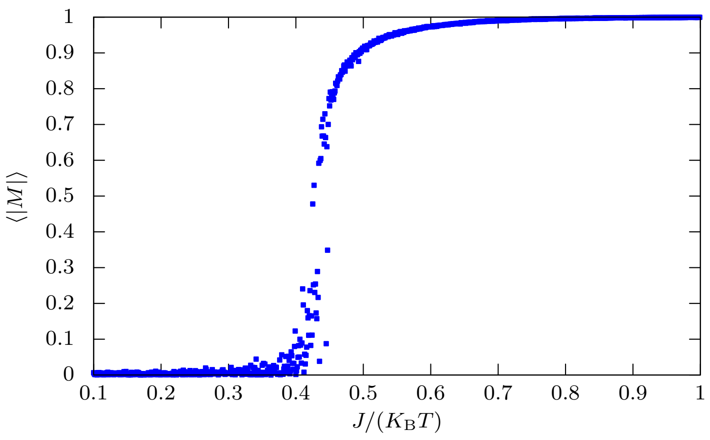

Figure 1: Absolute mean magnetization $\langle|M|\rangle$ versus effective temperature $\beta_\mathrm{eff} = J/(k_\mathrm{B}T)$. A phase transition occurs in the range $0.4 \leq \beta_\mathrm{eff} \leq 0.5$.

3.3 Comparison with analytical results

Compare your[5] estimated critical effective temperature $\beta^\mathrm{c}_\mathrm{eff}$ to the analytical prediction $J\beta_\mathrm{c} = \frac{1}{2}\ln(1+\sqrt{2}) = 0.4407$. Computing averages always allows one to introduce an error to your observable. Should one do it? Is the standard error a good choice?

[5]A run with $L = 30$, $N = 100$, 10 updates between every computation and a stepsize for the effective temperature of $\Delta\beta_\mathrm{eff} = 0.001$ should be sufficient to reproduce the theoretical prediction up to two decimal places.