Report for the Course Modelling in Computational Science, HT23

Project 1: The Potts model

Theo Koppenhöfer

Lund

September 15, 2023

Input : Initial data s, L, T, q

for $k = 0, 1, \ldots$ **do**
    Pick a point on the grid.
    Propose a new random spin value for this point.
    Calculate the change of energy ΔE that this spin flip would cause.
    Accept this spin flip with probability $\min\{1, \exp(-\Delta E/T)\}$.
end

Algorithm 1: Metropolis

Introduction

The following report is part of the first project of the Modelling in Computational Science course at Lund University, HT2023. In the first project we implemented a Monte-Carlo simulation of the q-state Potts model. The report and the Python implementation can be found online under [1]

The Potts model

A state of the q-state Potts model of a $L \times L$ flat grid is given by a mapping

$$s \colon \{1, \ldots, L\} \times \{1, \ldots, L\} \to \{1, \ldots, L\}\,.$$

One assigns this state an energy via

$$E = -J \sum_{\substack{i,j \text{ neighbouring} \\ \text{grid points}}} \delta(s_i, s_j)$$

where δ denotes the Kronecker-delta and $J = 1$ is the coupling strength. As given in the problem setting we assume periodic boundary conditions on the grid.

The algorithms

The Metropolis

The steps for the Metropolis algorithm are given in figure 1. A single iteration of this algorithm is performed by the function `MC_step_fast` in the implementation.

The heat-bath algorithm

The steps of the heat-bath algorithm are given in figure 2. In the implementation a single iteration is performed by the function `Gibbs_step`.

Input : Initial data s, L, T, q

for $k = 0, 1, \ldots$ **do**

Pick a point on the grid.

Pick a spin value for this point with a probability given by the distribution

$$p(s_i) = C \exp\left(\frac{1}{T} \sum_{j \text{ neighbours } i} \delta(s_i, s_j) \right).$$

Here C is a normalisation constant.

end

Algorithm 2: Metropolis

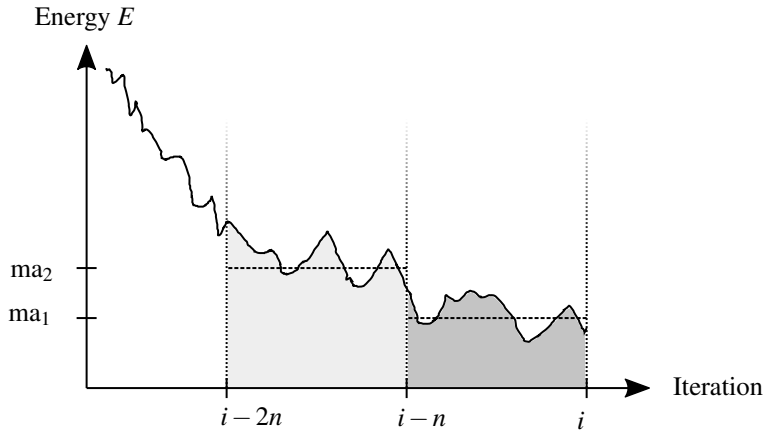


Figure 1: Explanation of the moving averages.

The implementation

Determining when the Energy has plateaued

The energy of the system takes some time t_0 to reach an equilibrium state. To determine t_0 we calculate in every step i moving averages ma_1 and ma_2 over n energies. The construction is shown in figure 1. If we start in the hot state then the energy will tend to shrink until we reach an equilibrium. Hence we can use the condition

$$\mathrm{ma}_2 \leq \mathrm{ma}_1 \tag{1}$$

to determine t_0. If on the other hand we start in cold state the energy will in general increase and we have to reverse the inequality in equation (1). After reaching an equilibrium the simulation runs for a further `M_sampling=5000` steps. It is over these samples we take the mean and the variance.

Improving performance

While simulating we run into performance issues due to the fact that Python is in general rather slow even with heavy use of Numpy. We resolved this issue with the help of Numba which precompiled functions and thus increased performance substantially. The downside is that the code becomes quite unestetic because Numba only supports some Python and Numpy features. In hindsight python seems to have been

The experiments

Distribution of energies in equilibrium

In the following numerical experiment we set the grid size to $L = 300$, the spin states to $q = 10$ and the temperature to $T = 10^2$. We then run the simulation until the system reached equilibrium. After that we varied the number of steps M of the simulation and plotted the distribution of energies. The results can be seen in figures 2 to 5. One sees that as M increases the distribution approaches a distribution which looks like the Maxwell-Boltzmann distribution.

State plots

In figures 8 to 6 we can see that as the temperature increases the state of the system becomes more chaotic. For low temperatures there are larger patches of the same state which corresponds to a lower energy configuration. In the implementation one can see an animated version of the evolution of the system state.

Energies in dependence of the temperature

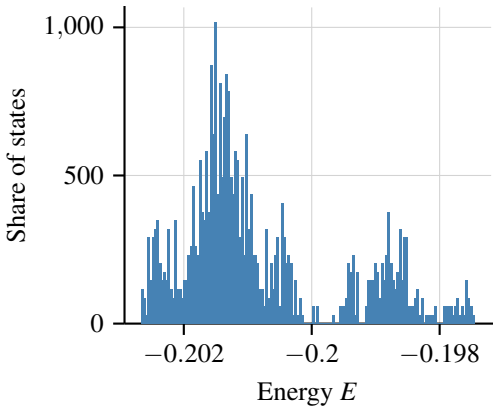


Figure 2: For $M = 10^5$

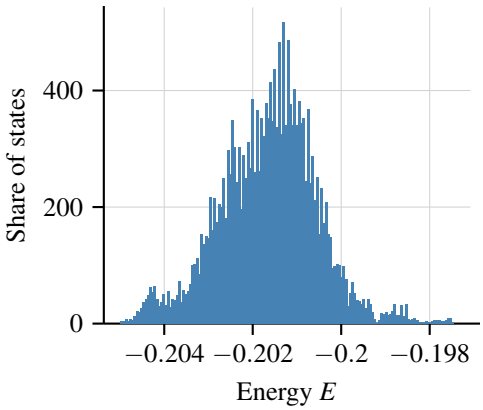


Figure 3: For $M = 10^6$

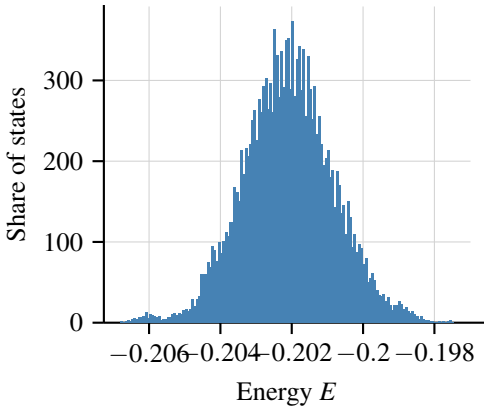


Figure 4: For $M = 4 \cdot 10^6$

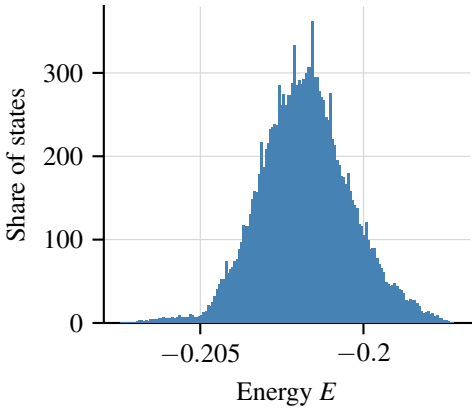


Figure 5: For $M = 10^7$

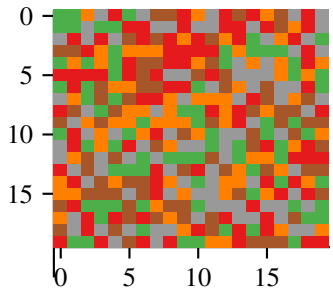


Figure 6: State for temperature $T = 0.1$.

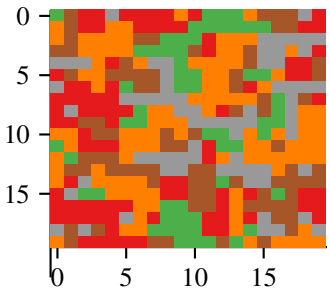


Figure 7: State for temperature $T = 1$.

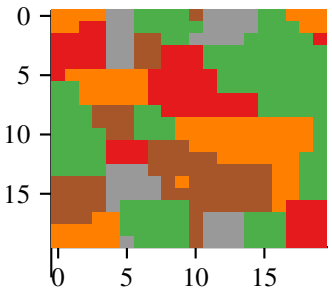
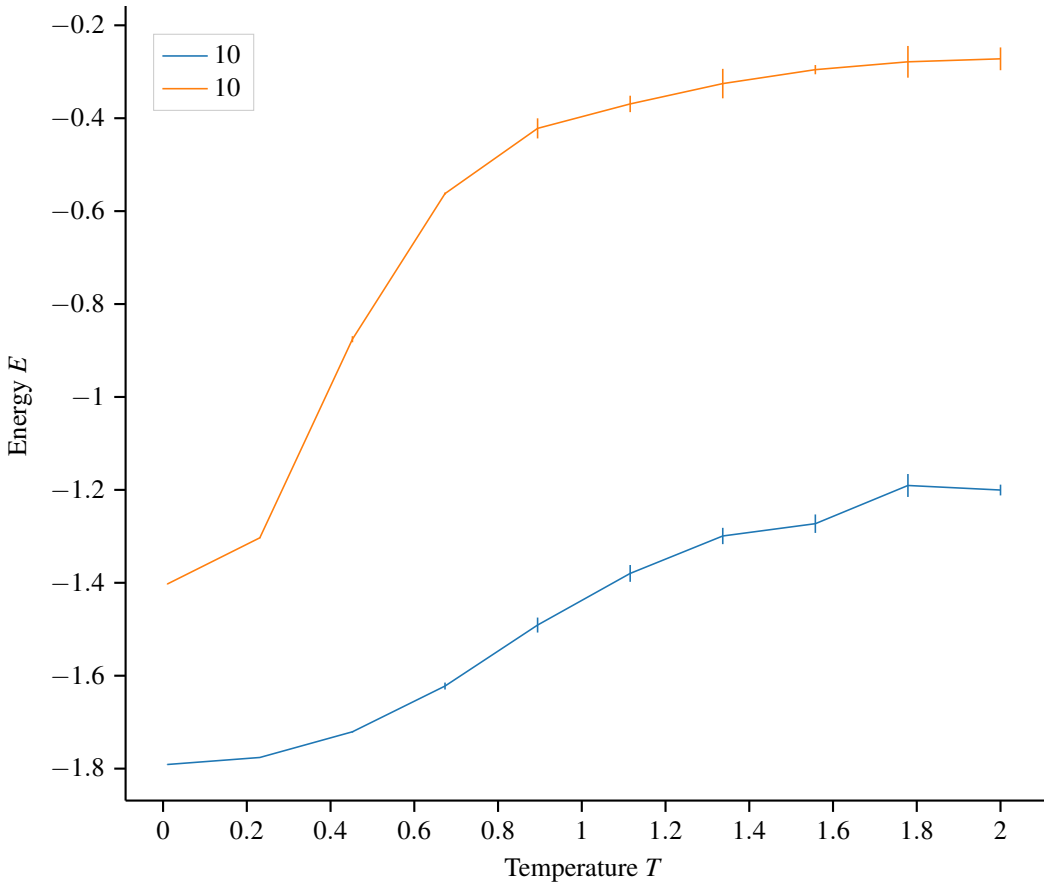


Figure 8: State for temperature $T = 100$.


10
10
−0.2
−0.4
−0.6
−0.8
−1
−1.2
−1.4
−1.6
−1.8
Energy E
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Temperature T

TODO:

- explain Metropolis algorithm
- explain Heat-bath algorithm
- briefly explain implementation
- sanity test hot vs. cold start
- energy evolution for maxwell-distribution test
- play around with phase transitions
- phase transition test -¿ E-T plots and t0 plot
- distribution based on energy

Bibliography

[1] computational-science-HT23, *Github repository to the project.* Online, 2023. [Online]. Available: `https://github.com/TheoKoppenhoefer/computational-science-HT23`.