Assignment 3: The Potts model

Peter Reinholdt

December 11, 2017

1 Introduction

The Potts model is a generalization of the Ising model, where a Monte Carlo Markov Chain is simulated on a lattice. In this assignment, we will be concerned with spins residing on a square, two-dimensional lattice, with side length L and periodic boundaries.

1.1 Model definition

In the Ising model, each lattice site contains a spin that is in either a $\uparrow$ or $\downarrow$ state. In the Potts model, this is generalized to $q \in [1, ..., q]$ possible spin labels, with the energy of a particular configuration σ

$$E(\sigma) = \sum_i \sum_{j \in n_i} \left(1 - \delta_{\sigma_i \sigma_j}\right) = 4L^2 - \sum_i \sum_{j \in \left\{i+\hat{1}, i+\hat{2}, i-\hat{1} i-\hat{2}\right\}} \delta_{\sigma_i \sigma_j} = 2L^2 - \sum_i \sum_{j \in \left\{i+\hat{1}, i+\hat{2}\right\}} \delta_{\sigma_i \sigma_j}, \quad (1)$$

where the $\hat{1}$ and $\hat{2}$ operators represent going along one of the two directions on the lattice. We can see that the Potts model reduces to the Ising model when $q = 2$, as there are then only two spin labels, equivalent to the original $\uparrow$ and $\downarrow$ spin states. Note how we can write the energy out in two ways: one considers explicitly all interactions between the spins; the other recognizes that the $i \to j$ interaction is the same as the $j \to i$ one.
The Potts model systems undergo a transition from ordered to disordered states when the temperature is lowered across the critical temperature. Where the Ising model undergoes a second order transition at the critical temperature, the Potts model with $q = 20$ instead undergoes a first order transition. This transition is expected to be visible from two distinct distributions in the energies, which occur due to phase coexistence. For the Potts models, the critical temperature depends on the value of q and is given as

$$\beta_c = \ln\left(1 + \sqrt{q}\right) \quad (2)$$

For the Ising model, an obvious order parameter is the average magnetization,

$$\langle |m| \rangle = \left\langle \left| \frac{1}{L^2} \sum_{i=1}^{L^2} \sigma_i \right| \right\rangle, \quad (3)$$

where $\langle \cdot \rangle$ denotes an ensemble average. This is a good observable when there are only two spins. With q spins, this is more difficult, but we could define an order parameter as

$$\langle |m| \rangle = \left\langle \left| \sum_{i=1}^{L^2} \exp\left(\frac{2\pi\sigma_i}{q}\right) \right| \right\rangle. \quad (4)$$

We see clearly that in the Ising case ($q = 2$), this just recovers (3). Secondly, if all spins are aligned, the order parameter will be 1, while the sum of the complex numbers will even out in a disordered state. Another good observable is of course the energy in (1), which will be used throughout.

Several methods can be used to simulate the Potts model. Perhaps simplest is the Metropolis algorithm, in which a change to the configuration of the system is *proposed* uniformly, with the proposal being accepted or rejected with a certain probability.
A more sophisticated algorithm is the Cluster algorithm, in which an inital seed site is selected randomly and uniformly on the lattice. A new spin value is drawn for this site from the remaining $q-1$ other values. Following this, the cluster is allowed to grow by adding neighboring spins of the same direction as the original seed. Cluster growth only happens with some probability $p_{add} = 1 - \exp(-\beta)$. After the cluster has grown to its final size, all the spins of the cluster are changed to the new spin value.

1.2 Autocorrelations

The important difference between the two methods is their auto-correlation times. The auto-correlation function is defined as

$$C_{xx}(t) = \langle (x(t_0) - \bar{x}(t_0))(x(t_0+t) - \bar{x}(t_0+t)) \rangle , \tag{5}$$

possibly including some sort of normalization. This measures the degree with which the observable x at time t_0 is correlated with an observable at time $t_0 + t$. If it is normalized, perfectly correlated observations have $C_{xx}(t) = 1$, while perfectly uncorrelated observations have $C_{xx}(t) = 0$. The integrated auto-correlation time,

$$\tau_{xx}(t_{\max}) = \sum_{t=1}^{t_{max}} C_{xx}(t) \tag{6}$$

gives a measure of the typical time after which the observations are no longer correlated.

1.3 Pseudorandom number generators (PRNGs)

Computers are deterministic machines, so generating random numbers on a computer is difficult. We can, however, generate sequences of numbers that, while deterministic, satisfy many of the properties of a true random distribution. A wide range of PRNGs exist, with variying quality in the generated pseudo-random numbers and in their speed. Both points are crucially important, the first one since bad random can seriously deteriorate the quality of Monte Carlo simulations (see ex.); the second one owing to the fact that a significant fraction of the computational effort spent in performing Monte Carlo simulations is spent generating random numbers. For the purpose of the Monte Carlo simulations we will be investigating, the `xoroshiro128+` (see ex. `http://xoroshiro.di.unimi.it/` for a discussion) algorithm will be used. The algorithm is based on work by Marsaglia. This PRNG is very fast and passes the TestU01 tests with no systematic failures (unlike for example `MT19937_64`, which systematically fails the `LinearComp` test).
As the name implies, `xoroshiro128+` contains 128 bits of internal state, and applies a sequence of bitwise `xor, rotate` and `shift` operations in order to generate new pseudo-random numbers.

2 Implementation and testing

The source code is available on GitHub, see
`https://github.com/peter-reinholdt/MM837/tree/master/Assignment_3`. The project is build with CMake, and the compilation process can be summarized in the following steps:

```
git clone https://github.com/peter-reinholdt/MM837.git
cd MM837/Assignment_3/
mkdir build
cd build
```

```
cmake ..
make
```

After which the program can be executed by passing an input-file to the executable

```
./lattice settings.inp
```

The input file contains adjustable parameters of the simulation, like system properties, the kind of method to use to explore the system and output settings.

2.1 Code organization

The input file is a `toml` file, which is parsed with the help of an external, header only C++11 library (see `https://github.com/mayah/tinytoml`).
The code is organized into two main folders, `src`, and `include`, which contains the main source files and header files, respectively. The source file are organized in logical units:

```
include/
├── properties.h
├── solvers.h
├── statistics.h
├── toml.h
└── xoroshiro128plus.h
src/
├── main.cc
├── properties.cc
├── solvers.cc
└── xoroshiro128plus.cc
```

The main program is responsible for the input processing, and also acts as a driver program. The main routines for simulating the systems in contained in `solvers.cc`, while `properties.cc` contains routines to calculate energies, auto-correlations and so on.
The file `xoroshiro128plus.cc` implements the `xoroshiro128+` PRNG. It is implemented as conforms to the standards of a `C++ UniformRandomBitGenerator,` i.e., it exposes methods for telling the type of the number returned by the generator object (`UINT64_T`), as well as the smallest and largest possible values that the generator can return, and an `operator()` that actually provides the random numbers when called. By doing this, we can pass `xoroshiro128plus` generator object to a `std::random` distribution in order to conveniently generate random numbers according to some desired distribution.

2.2 Program input and output

The program parameters is specified in a settings-file. A typical input looks like

```
[[solver]]
solver_type = "metropolis" #or cluster or hybrid
n_steps_therm = 10000
n_steps_prod = 10000000
metropolis_freq = 10 #only used with the ‘‘hybrid’’ solver

[[system]]
side_length = 200
potts_q = 2
```

```
beta = 0.88137358702
#beta = 1.69966902559

[[output]]
outfile = "output.dat"
outfreq = 1000
conf_outfreq = 1000
```

In the example above, as simulation with $q = 2$ spin values is requested on a 200×200 lattice at the critical temperature $\beta_c = \ln\left(1 + \sqrt{2}\right) = 0.88137...$, using the Metropolis algorithm.
As a result of the simulation, the program outputs energies measured every `outfreq` (1000) steps to a file `output.dat.energies`, and calculates and stores autocorrelations and auto-correlation times in the file `output.dat.autocorr`. Further, every `conf_outfreq` (1000) steps, configurations are dumped for visualization.
Energies are not tracked as the simulation proceeds, but are instead only calculated "the hard way" when requested. The calculation of the energy does require an amount L^2 of work, but since it is only done periodically, this is not really a dramatic cost. Further, the computational cost is dominated by the generation of pseudorandom numbers, and the cost of totally recalculating the energy (perhaps even at every macro-iteration) is basically negligible.

2.3 Metropolis algorithm

The metropolis update consists of the following steps

1. From an initial configuration, σ, propose a new configuration σ'.
2. Evaluate the difference in energies between the configurations, $\Delta E = E(\sigma') - E(\sigma)$
3. If $\Delta E \leq 0$, accept the change.
4. Otherwise, accept the change with probability $\exp(-\beta \Delta E)$
5. Go to 1.

The selection of a new configuration can be done in multiple ways, but it should be done uniformly in all cases. This can be done with a typewriter method, e.g. by going through the sites sequentially from top to bottom; by picking a random site uniformly; or by picking half of all sites at ones in a checkerboarding fashion.
For the typewriter method, there are only 8 possible values of ΔE, so the computational cost of the exponential can be removed by keeping these values pre-calculated in an array. It is pointless to propose the same spin in an update, so one should make sure that the new selected spin is different from the first one. This can be done by drawing a random integer from $q - 2$ values. In the case that the spin is the same as the original one, the spin is instead assigned the maximal value. This ensures that a new spin is picked uniformly from the remaining $q - 1$ spin labels.

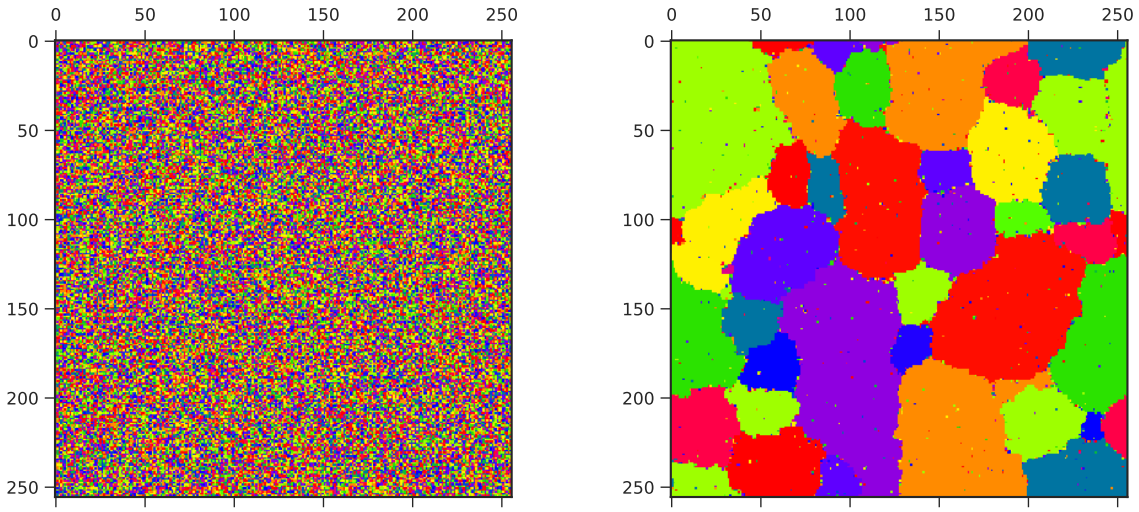
2.4 Wolff cluster algorithm

A sweep of the algorithm proceeds as follows:

1. Pick a random site uniformly on the lattice
2. Choose a new value for the spin; immediately change the spin to this new value
3. Add neighbouring spins of the same direction as the original spin direction to the cluster with probability $p_{add} = 1 - \exp(-\beta \Delta E)$.
4. Go to 2.

2.5 Testing

In order to test the correctness of the program, it is useful to consider the limiting regimes as $\beta \to \infty$ (low temperature, ordered) and $\beta \to 0$ (high temperature, disordered). Figure 1 gives a typical configurations for the $q = 20$ model for above and below the transition temperature. We see clearly the expected behavior: as the temperature becomes large, the many configurations energy penalty in going to a high energy is small compared to the temperature, so the many disordered configurations becomes populated, and the configuration is essentially random. At low temperature, the opposite is true, and the energetic advantage of spins pointing in the same direction wins out, and we see domains of like spins forming.



(a) A disordered state, with $\beta = 0.5$.

(b) An ordered state with $\beta = 2.0$.

Figure 1: A sample configuration of the Potts model, $q = 20$, on a `256x256` lattice, simulated above and below the critical temperature. The color encodes the different values of σ_i at a particular position on the lattice.

In order to be more sure of our implementation, we can also test that the transition from disordered to ordered states actually occurs at the predicted $\beta_c = \ln\left(1 + \sqrt{q}\right)$. To do this, the average energy was tracked as a function of the temperature. The result of this is shown in figure 2. From this figure, we can see that at theoretical value of β_c, a dramatic and sudden change in the average energy also occurs, which is in line with a phase transition occuring. In conclusion, the implemented model have many of the expected behaviours, so we can be reasonably confident that the implemetation is correct.

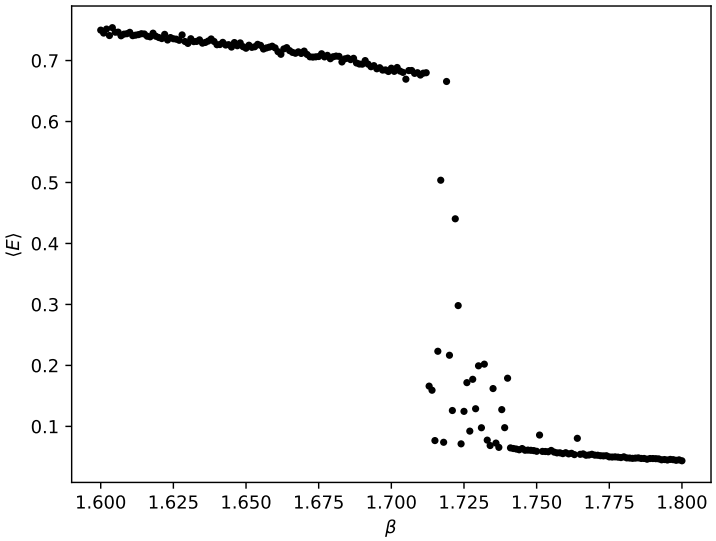


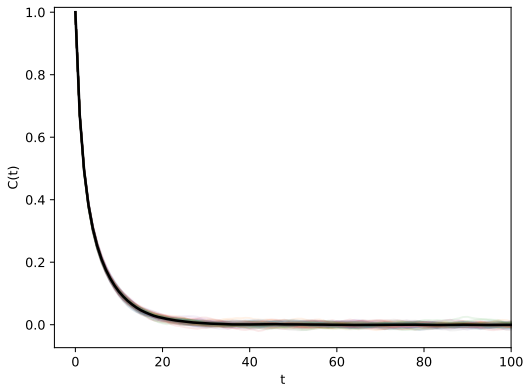
Figure 2: Phase transition in Potts $q = 20$ model on a 20x20 lattice. The average energy is plotted as a function of the temperature. The model was simulated with the cluster algorithm, with 10000 thermalization steps, and 100000 production steps, measuring energies every 100 steps.

3 Results and Discussion

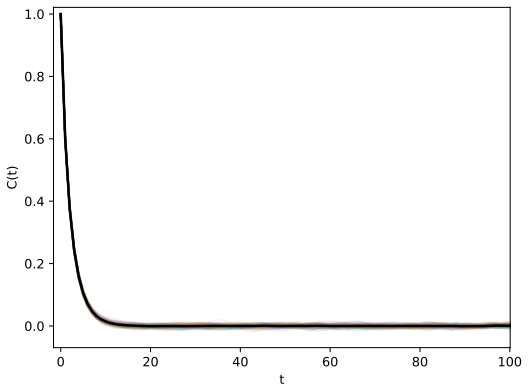
In this section, the autocorrelations of the Potts model with $q = 2$ and $q = 20$ will be discussed. A lattice size of 12x12 is used for investigating the autocorrelation times. A lattice size of 20x20 is used for the generation of energy distributions.

3.1 Autocorrelation times

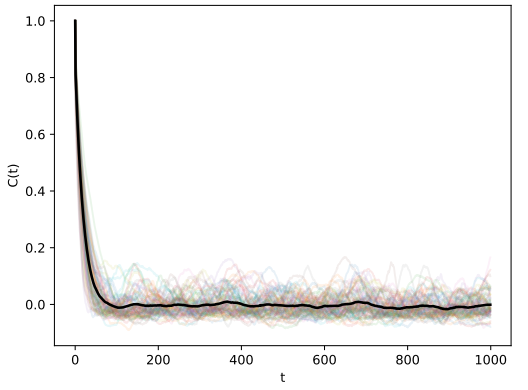
The autocorrelation functions of the Potts model simulated with either the Metropolis or the Cluster algorithm is shown in figure 3, and the integrated autocorrelation function is shown in figure 4. For $q = 2$, 10^5 steps were used for both the thermalization and the production run, with measurements taken every step. For the $q = 2$ model, 10^6 steps were used for thermalization and 10^7 steps for the production. In both cases, measurements were performed every 1000 steps. The systems were all simulated at their critical temperatures.



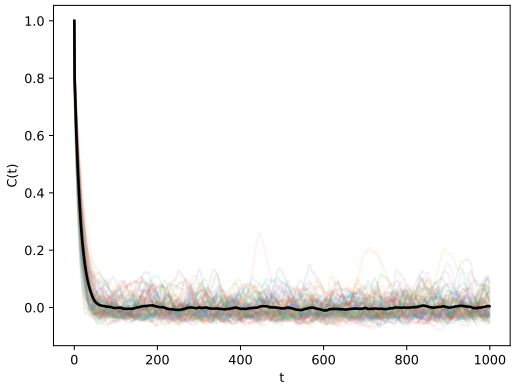
(a) Metropolis algorithm, $q = 2$.



(b) Cluster algorithm, $q = 2$

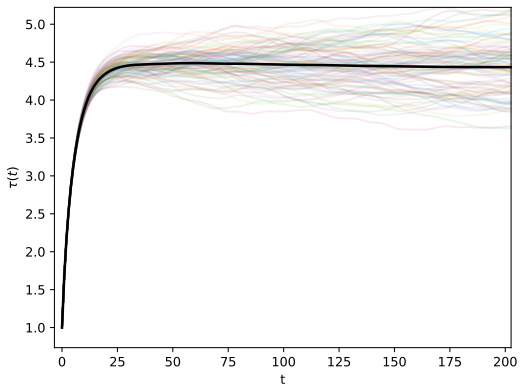


(c) Metropolis algorithm, $q = 20$.

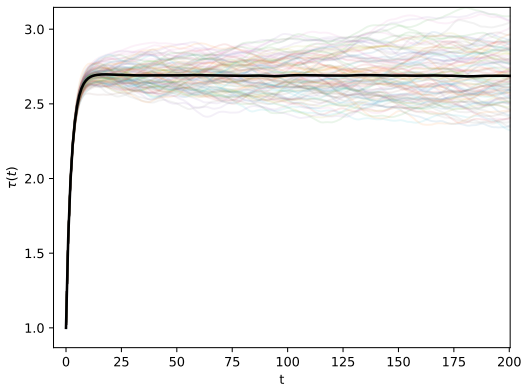


(d) Cluster algorithm, $q = 20$.

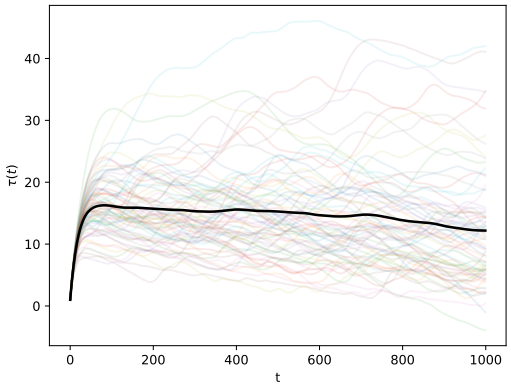
Figure 3: Autocorrelation function of the Potts model. For $q = 2$, the autocorrelation is shown for 100 individual runs as thin lines; for $q = 20$ the time is in units of 1000. The average of the measurements is shown as a bold line.



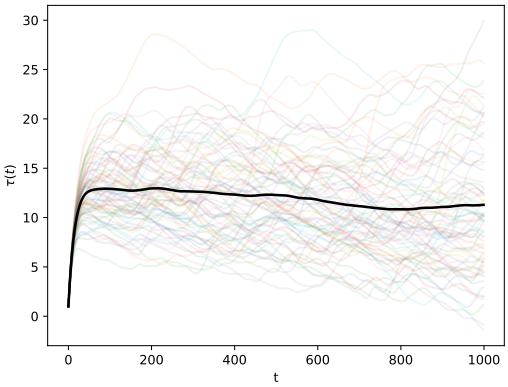
(a) Metropolis algorithm, $q = 2$.



(b) Cluster algorithm, $q = 2$



(c) Metropolis algorithm, $q = 20$.

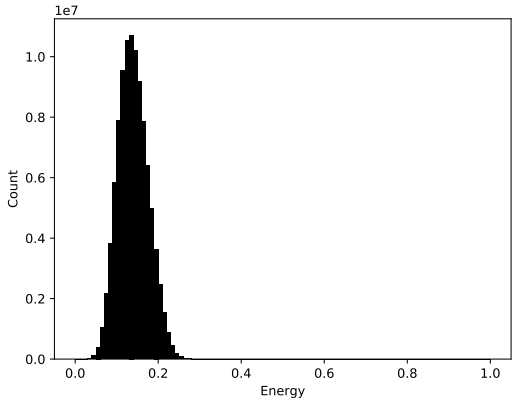


(d) Cluster algorithm, $q = 20$

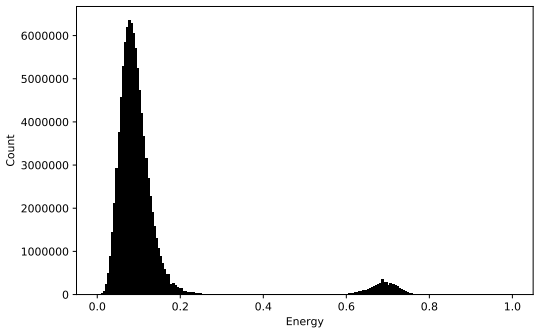
Figure 4: Integrated autocorrelation function of the Potts model. The autocorrelation is shown for 100 individual runs as thin lines, and the average of the measurements is shown as a bold line.

3.2 Energy populations

The energy populations of the Potts $q = 2$ (Ising) and $q = 20$ models are shown in figure 5. The systems were run with the cluster algorithm. The $q = 2$ system was thermalized for 10^5 steps, while the $q = 20$ system was thermalized for 10^6 steps. The systems were both run for 10^8 steps, with measurements of the energy being done at every step. A trace of the energy in the two cases is shown in figure 6.

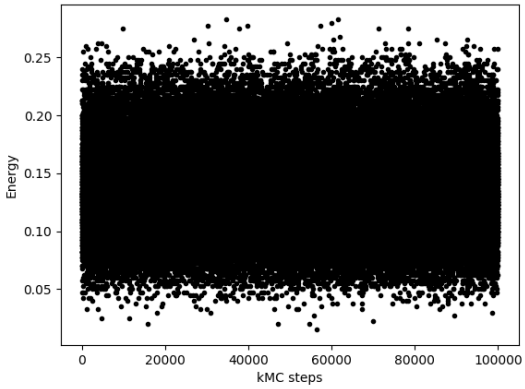


(a) Distribution of the energies of the $q = 2$ Potts model.

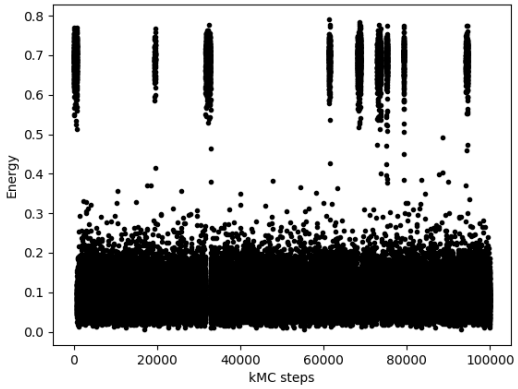


(b) Distribution of the energies of the $q = 20$ Potts model.

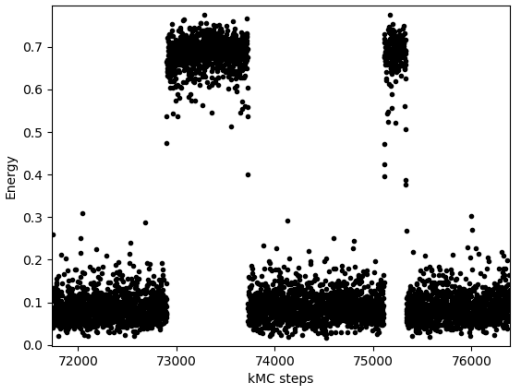
Figure 5: The energy populations of the Ising and Potts $q = 20$ models.



(a) A trace of the energy in the$q = 2$ Potts model.



(b) A trace of the energy in the Potts $q = 20$ model.



(c) A zoom on a part of the trace of the energy in the $q = 20$ model.

Figure 6: Traces of the energy in the Potts model.

3.3 Hybrid model

Table 1 shows the effects of periodically mixing in a metropolis step to the cluster algorithm, that is: after doing n cluster sweeps, a single metropolis sweep is also performed. For this, the $q = 20$ Potts model was used.

Typewriter frequency	τ_{int}
1	8000-10000
2	12000
3	10-14000
Metropolis	16000
Cluster	13000

Table 1: The table shows the integrated autocorrelation time, as well as acceptance ratios for various hybrid models, as well as the standard Metropolis and Cluster algorithms.

4 Analysis and Conclusions

4.1 Autocorrelations

We see that the integrated autocorrelation decays, as expected, according to an exponential. The importance of sampling is, however, also clear from figure 3, as the measurements of individual runs can deviate quite significantly from the averaged value of 100 runs, especially for the $q = 20$ Potts model. Looking at the integrated autocorrelation times in figure 4, we see that the times are in general much longer for the $q = 20$ case than in the Ising case of the Potts model, for both of the algorithms tested. The cluster algorithm shows slightly better performance in terms of the autocorrelation times; the performance is perhaps even better, as the work required per iteration is lower than in the Metropolis model, since only a subset of the L^2 spins need to be considered, and, followingly, that we need to generate fewer random numbers per iteration.

4.2 Energy populations

We see clearly the differences between the Potts $q = 2$ and $q = 20$ models from the energy traces in figure 6. Whereas there the energy in the Ising case fluctatates around only a single mean energy, in the $q = 20$ case, there are two clear and distinct populations, at completely different energies. Once in a while, we see that the energy switches between these two phases. This is evidence of phase-coexistence, and it is also clearly seen from figure 5 that there are indeed two distinct distributions in the $q = 20$ case, whereas there is only a single one in the Ising case. The existence of the two peaks in the high-q Potts model points to this system having a first-order phase transition at the critical temperature.

4.3 Hybrid methods

From table 1, we see that mixing in a metropolis update every few steps can be slightly beneficial to reducing the autocorrelation of the cluster algorithm, but the mixing in of the metropolis updates doesn't appear to change reduce the autocorrelation time hugely.

5 Conclusion

The Potts model has been simulated using both the Metropolis and the Cluster algorithm. The autocorrelation times of the Cluster algorithm were found to be significantly lower than the standard Metropolis, and further small improvement was demonstrated with a hybrid algorithm. The autocorrelation was found to be much higher with a high number of possible spin labels ($q = 20$) than in the Ising case ($q = 2$). The $q = 20$ Potts model was shown to exhibit an interesting behaviour at the critical temperature, with evidence of multiple phases coexisting, something which was not present in the Ising case.