Coding the Model

My approach to implementing the Ising model involved the use of a `Lattice` struct, as defined in the header file, in order to ensure modularity and to make the code easily extendable. The basic usage involves instantiating a `Lattice` using the `new` function, running the simulation using the `run` function, and thereafter performing calculations on the modifed `Lattice` pointer to obtain correlation time, specific heat, etc.

The functions `phase_diagram`, `autocorrelation`, `plot_specific_heat`, and `write_spins` implement helper functions to output data for different temperatures and lattice sizes which may in turn be plotted using Python scripts. These were used to create the plots on the following pages. Note the use of `#pragma omp parallel for` preprocessor directives to parallelize the execution of these functions, which significantly decreased their runtimes.

The `main` function may easily be modified to implement new functionality using the existing helper functions, or to reproduce results.

All code and plots maybe be found on GitHub at: https://github.com/niknaknel/ising

Autocorrelation

Figures 1 (below) and 2 (next page) were produced by generating data using the `autocorrelation` function and passing `TRUE` to the call to `correlation_time`, which was then plotted using Python.

In figure 2, the vertical red line on each plot indicates the correlation time determined by the program, which is taken to be the first time step at which the autocorrelation function falls to 1/e of its initial value. These plots are similar to figure 3.5 in Newman in that they depict approximate exponential decay of the autocorrelation function over time.

Figure 1 depicts the mean correlation time of 10 measurements at 25 different temperatures. It resembles figure 3.8 in Newman and illustrates the effects of critical slowing down, i.e. the spike in correlation time around the critical temperature.

Equilibration

The following two pages contain plots depicting how the program determines equilibration time for (i) temperatures far from the critical temperature, and (ii) temperatures near the critical temperature.

The equilibration time, indicated on the plots by the red vertical line, is determined by the `equilibration_time` function as follows:

1. Initialize two lattices of the same size with different initial states. M1 is initialized at zero temperature (spins are aligned), while M2 is initialized at infinite temperature (spins are randomly initialized).
2. Run the simulation for each lattice for 10 000 time steps.
3. Compare the difference in magnetisation per site for each time step. If enough consecutive differences are within a certain threshold, equilibrium has been reached.

(i) There appears to be no clear dependence of equilibration time on lattice size. However, for small lattices there is more variability in magnetisation per site simply due to division by a small N value (see diagrams for T=3.5). Thus, the algorithm adjusts the difference accordingly.

(ii) Near the critical temperature there is also more variability in magnetisation, i.e. it undergoes critical fluctuations, as described in Newman. Therefore the algorithm increases the size of the threshold to compensate for this. However, the equilibration time tends to be longer and not as well-defined in this region.

The values for THRESHOLD and DIFF_MAX were adjusted until the function consistently returned reasonably accurate values for equilibration time.

Phase Diagram

The plots to the left (figure 5) were produced by running the `phase_diagram` function for lattice sizes of 50×50 and 100×100, respectively, then plotting the output using Python. The plots depict the grand mean magnetization per site of 10 sample means at 25 different temperatures. In each sample, the magnetization was measured 10 times every 2τ steps, where τ is the correlation time, starting at equilibrium.

The phase diagram adheres to the expected behaviour of magnetization for the the given temperatures. However, the values recorded for L=100 are closer to theoretical predictions and are less error prone due to greater stability of magnetization for larger lattices. Also visible is the increased error margin in the critical region (near T=2.2). The figure clearly depicts the phase transition in the critical region, and confirms our choice of critical temperature.

Specific Heat

The plot below was generated by running the `plot_specific_heat` function for a lattice of size 100×100 and then plotting the output using Python. Each point represents the mean specific heat of 10 measurements at the given temperature. In each case the specific heat was calculated using equation 3.15 in Newman, and by sampling the energy of the lattice at equilibrium 10 times.

Although the curve takes on approximately the expected shape (as in figure 3.10 of Newman), the measurements were clearly innacurate and extremely error prone. This is most likely due to the small sample size of recorded energies when calculating each specific heat value. The measurements could be made more accurate through the use of a much larger sample size or sampling method such as bootstrapping or jackknifing.

Animating the Lattice

The figure to the right shows the equlibration of a 100×100 lattice at a temperature of T=1.5 over time. The plots are stills from an animation created using the `write_spins` function and a Python script. The lattice is initially at infinite temperature (1) and changes state until it has equilibrated (8). The red pixels represent spin up, while the green pixels represent spin down. The corresponding video may be found in the `out/spins/` directory in the project files.

Improvements

The model could be made more accurate in some respects. This might be done by,

- treating the critical region differently when taking measurements to decrease error margins. Modifications to the algorithm to accommodate this are alluded to at the end of chapter 7 in Newman.
- implementing a more accurate method to determine equilibration time.
- implementing more robust sampling techniques when calculating values such as specific heat or magnetic susceptibility, e.g. bootstrap, jackknife methods.

Furthermore, the model could be made more comprehensive by testing larger lattice sizes and extending its functionality.