Ising Model Readme

Anthony Bourached

March 27, 2015

1 Readme

This is a short document to answer any likely questions that may arise about the accompanying codes. Also contains some notes on how to run. Both programs output to a file 'netmag' (as at first i was only measuring magnetisation) with four columns:

1. β
2. Timestep Number
3. Magnetisation
4. Energy

Many further codes were written to analyse the above data. All of these were written in Python. Please contact me for the code for any specific part of the project.

2 Metropolis

2.1 About the code

This code was written in c. I originally wrote this for an assignment for Computer Simulation taught by Prof. Tom Archer in the previous year. For this project I made major alterations to the code mostly with the aim of reducing clutter in main by performing most necessary sub-tasks in functions such as direction(), which took an interger as an argument 1, 2, 3 or 4; corresponding to left, right, up and down and returned the spin at that site. A more significant adjustment to the code was random number generator, which I had seeded with time in my previous program. In this code I used Martin Lüscher's random number generator 'ranlux' [1] which is reliable for large Monte Carlo Simulations. This gives a convenient way of choosing our seed which was particularly inportant in debugging as well as determining thermalisation.

2.2 Using the code

All parameters of our simulations can be entered through the terminal when we run the code except for L which can be entered at the top of the program. The user interface follows the following structure:

1. Initial start state: enter '1', '2' or '3'. Old state start was not used for the project but may be useful if we want to produce an observable as a function of temperature but want to minimise timesteps.
2. Timestep Number: This is the number of Monte Carlo Steps desired for each temperature in range (see later)
3. Step Size: We run the code from a minimum inverse temperature to a maximum one in steps of step size 'step size'.
4. Minimum Temperature
5. Maximum Temperature

3 Wolff Cluster

3.1 About the Code

This code was started in Michaelmas term of this year specifically for this project. In Hilary term it became apparent that it was not functioning properly so analysis had to continue on the Metropolis algorithm while the problem was investigated. Finally this code was re-written in c++ as I was using this language much more and found it much easier. This is the code provided.

3.2 Using the Code

All parameters of the simulation can be entered through the terminal when we run the code.

1. Enter L.
2. Initial start state: enter '1' or '2'.
3. Timestep Number: Number of Monte Carlo steps.
4. It was found to be less cumbersome not to include the step size and temperature range in cout. These may be altered from main where illustrated.

References

[1] Martin Lüscher, User's guide for ranlxs and ranlxd v3.2, (December 2005).
`http://luscher.web.cern.ch/luscher/ranlux/guide.pdf`

[2] Konstantinos N. Anagnostopoulos, Computational Physics (National Technical University of Athens, 2014).
`http://www.physics.ntua.gr/ konstant/ComputationalPhysics/Book/ComputationalPhysicsKNA.pdf`