Course: Molecular Modeling, 2025/26
Part II: Monte Carlo Methods

Computational Project

Equilibrium Monte Carlo simulation of the 2D Ising model

Important: please read very carefully this document, and follow closely the submission instructions at the end.

A. Introduction

The Ising model is a milestone of statistical physics. In a famous mathematical tour de force, in 1944 Lars Onsager was able to solve exactly the model in the $L \to \infty$ limit in two dimensions. The Onsager solution exhibits a continuous phase transition from paramagnet to ferromagnet (or liquid-gas transition in the lattice-gas interpretation) at the critical temperature $T_c = 2/\ln(1+\sqrt{2}) = 2.269185...$. Subsequently, Ferdinand and Fisher computed exactly the free energy in 2D for finite-size lattices.

The goal of this computational project is to implement a Markov-chain Monte Carlo code for the 2D Ising model in the canonical ensemble, and two simple codes for the statistical analysis of the data, binning and (optionally) jackknife. You will first check your code by comparing your data with the Ferdinand-Fisher exact result (a necessary test for the correctness of the code) for small system sizes, and then perform simulations with larger sizes at different temperatures, in order to see the different behavior below, above, and near the critical temperature.

While the Ising model simulation is relatively easy to implement, it allows to gain familiarity with the most important general aspects of Monte Carlo simulations in systems with many degrees of freedom, such as equilibration, autocorrelation, phase coexhistence, phase transitions, and critical slowing down. In addition, the availability of exact results allows to detect possible mistakes in the code or the data analysis. Students who will enroll in the course "Advanced Simulation Methods" in the second semester will reuse the codes developed in this project, adapting them to more advanced Monte Carlo algorithms and/or extensions of the model.

B. Required tasks

1. Implement the Metropolis algorithm (with single-spin flip Glauber dynamics and random updating) for the Ising model without magnetic field, defined by the energy function $E = -\sum_{\langle i,j\rangle} S_i S_j$, on the two-dimensional square lattice of linear size L with $N = L^2$ spins, using periodic (toroidal) boundary conditions and nearest-neighbour interactions. The code must have the following input/output:

   Input:

   - L (linear size of the lattice),

- T (temperature),
- n_{MCS} (total number of MCS, where 1 MCS = N attempted spin flips). (I repeat: 1MCS = N attempted spin flips, not **one** attempted spin flip!)
- n_{meas} (number of MCS between two successive measurements), seed of the random number generator.

Note: L can either be specified at compile time (for example with a Makefile) or at run time (in the latter case, dynamical allocation should be used, in order not to waste memory).

Output: time series for the magnetization $M = \sum_{i=1,N} S_i$ and the energy E every n_{meas} MC steps (length of the time series = n_{MCS}/n_{meas}).

The program must be structured with the following separate functions (or subroutines in Fortran):

a) input (which can be contained in the main program)

b) output

c) initialization of the lattice (geometry);

d) Monte Carlo update

e) measurement of the observables M, E.

The Monte Carlo update should be independent of the geometry, i.e. it should work also in three dimensions, for a triangular lattice, for the square lattice with the next-nearest neighbors, etc. The only information about the geometry of the lattice should be contained in an array listing the neighbors of each site. This array should be created in the function c) above.

2. Report the speed of the Monte Carlo update, expressed in number of attempted spin flips per second.

3. Write a code that groups the data in a time series into “bins” of size m and computes the statistical error of the average of the binned data, for different values of $m = 1, 2, 4, 8, 16, 32, \ldots$ (the largest value of m should be such that there are of order 100 bins).

4. To test the correctness of your MC code, perform a simulation with $L = 20$, $T = 2.0$, $n_{meas} = 10$ starting with random initial conditions. Try at least with $n_{MCS} = 10^8$ (this allows to estimate $\langle E \rangle / L^2$ with a relative statistical error of less than 10^{-5}).

   Discarding the first 10^3 MCS, compute the sample average $\overline{E}$. Use your binning code, plot the statistical error of $\overline{E}$ as a function of m. From this, report your final estimate of $\langle E \rangle / L^2$ **with its statistical error** (standard deviation).

   Comment on whether your estimate agrees, **within its statistical error**, with the *exact* value given by Ferdinand and Fisher $\langle E \rangle / L^2 = -1.7455571250...$ for $L = 20$ and

$T = 2.0$. Agreement is necessary for your code to be correct. The larger n_{MCS} the more stringent the test will be. If there is still no agreement after carefully checking the code, try with a different random number generator. If this still fails, contact me. You can also try different temperatures.

5. After the code has passed the Ferdinand-Fisher test above, perform "production runs" for $L = 100$, and at least these three values of the temperature: $T = 2.0$, $T = 2.27$, $T = 2.6$, starting from a random initial configuration. Use $n_{meas} = 10$ and at least $n_{MCS} = 10^6$ (for compiled codes, $n_{MCS} = 10^8$ should be possible). If you have sufficient computer time, you can try to do the simulations for $2.0 \leq T \leq 3.0$ in steps $\Delta T = 0.1$.

6. Plot the time series for E and M (not $|M|$!) for the three temperatures above for the first 10^3 MCS, and for the first 10^6 MCS (in this case plot no more than one point every 100 data so you don't have too many points). Give your interpretation of these plots. Plot the time series for $L = 20$ for M (not $|M|$!) at $T = 2.0$ for $n_{MCS} = 10^6$ (one point every 100 data) and comment on the difference with that for $L = 100$.

7. Use the binning code to compute, for the three temperatures above, $\langle E \rangle$ and $\langle |M| \rangle$, and the statistical errors of the binned data. Plot this error as a function of m for both quantities at each temperature. From this plot, obtain your final estimate of the statistical error on $\langle E \rangle, \langle |M| \rangle$, and estimate the autocorrelation time of E and $|M|$. Comment on the dependence of the autocorrelation time on temperature.

8. Plot $\langle E \rangle / N$ and $\langle |M| \rangle / N$, **with their statistical error**, as a function of the temperature. Give an intepretation of the plots.

Important: in all the plots, M and E should be normalized by N.

C. Optional tasks

1. Write a code that takes as input the time series of a given quantity X and returns the jackknife estimates and jackknife errors of the following quantities: $\langle X^2 \rangle$, $\langle X \rangle^2$, $\langle X^2 \rangle - \langle X \rangle^2$, (Suggestion: write the code so that you can later reuse the jackknife routine for *any* function $f(X, Y, ...)$ of any number of variable.)

   Use together your binning (using a value of m previously determined in Required task n.7) and jackknife codes to compute the following quantities and their statistical errors:

   - magnetic susceptibility per spin $\chi = \frac{\beta}{N}(\langle M^2 \rangle - \langle |M| \rangle^2)$
   - heat capacity per spin $c = \frac{\beta^2}{N}(\langle E^2 \rangle - \langle E \rangle^2)$

   Plot the jackknife error of χ, c as a function of m and from this estimate their statistical error. How does this error compare with the one you would obtain from "error propagation"? (answer to this question for at least one of these quantities and $T = 2.27$).

   Plot χ, c, with their statistical error, as a function of T and interpret the plots.

2. Include in your jackknife code the observables $\langle X^4 \rangle$, $\langle X^2 \rangle^2$, and "Binder parameter" $g_X = \frac{1}{2}(3 - \frac{\langle X^4 \rangle}{\langle X^2 \rangle^2})$. Repeat the previous task for the Binder parameter g_M and plot this as a function of T.

3. Repeat the production runs for other values of L (for example $L = 50, 200, 400$) and determine T_c and the critical exponents by perform a finite-size scaling analysis of the data, using the finite-size scaling forms

$$\langle |M| \rangle / N = L^{-\beta/\nu} f_M((T - T_c)L^{1/\nu}), \quad g_M = f_g((T - T_c)L^{1/\nu}),$$
$$\chi = L^{\gamma/\nu} f_\chi((T - T_c)L^{1/\nu}), \quad c = L^{\alpha/\nu} f_c((T - T_c)L^{1/\nu}),$$

where f_M, f_g, f_χ, f_c are suitable functions.

4. Plot snapshots of equilibrium configurations at different temperatures for $L = 100$ and/or histograms of the magnetization.

5. Using the time series produced in your production runs (after discarding a sufficient number of initial MC steps) compute the autocorrelation function $C_E(t)$, $C_M(t)$ of E and M and plot them.

6. Add a magnetic field and perform simulations at $T = 2.0$ and h between -1 and 1.

D. Important points

- The project should be done in groups of three (preferably) or two students.
- The computational project is graded up to 9 for doing the required tasks and up to 10 for doing also some of the optional tasks (n.1 is recommended).
- Start early, and if you get stuck, talk to me and to your classmates, don't wait until the final report.
- The programs should be written in C, C++, or Fortran. If you are only familiar with Java or Python, you can also use one of these two languages, provided the codes are *compiled*. Non-compiled Python codes will probably be too slow. Interpreted languages such as Mathematica or Matlab are not admitted, except for plotting and data manipulation.
- If you use any kind of artificial intelligence, you *must* clearly declare exactly what AI you used, and what you did with it.
- Use the random generators provided. Do not use intrinsic generators. Bad generators are known to produce incorrect results.

- For the Metropolis algorithm, create a *table with precomputed values* of $\exp(-\beta\Delta E)$ to avoid computing exponentials at every attempted spin flip, which is expensive.
- Try to optimize your code for speed. Usually 90% of the CPU time is spent in a few lines. To find those lines, you can use a tool like "gprof", or "time" instructions in the code, or common sense.
- Try to automatize repetitive tasks such as the sequence compile $\rightarrow$ run $\rightarrow$ analyse $\rightarrow$ produce plots. To this end you can use shell scripts (my favourite!), awk, Python, or similar tools. If you find yourself repeating the same task more than three times, you should automatize!
- Use the computing cluster responsibly: avoid filling the disk with huge files and running computations on the master node. Compress your large files when you are not using them. Consider writing in binary form to save space.

E. Submission instructions

There will be three practice sessions. You are asked to submit partial results **in class, at the end of each session**, mainly to check that you are making good progress.

Important:

- Each submission should be uploaded as single gzipped tarfile to the Campus Virtual.
- The file name should be LastnameLastname2Lastname3practice1.gzip
- Provide only one submission per group.

After session n.1 (send at least one day before session n.2):

- Your code that initializes the system, creates the 2D geometry and computes the energy and magnetization [functions a),b),c),e) in point 1 of the Required Tasks].
- Energy and magnetization of the test configuration and the "mistery" configuration.

After session n.2 (send at least one day before session n.3):

- Your complete Monte Carlo code and the binning code.
- Plots of the square root of the sample variance of the binned data, $s_{\overline{X}^{(m)}}$, as a function of m for the test distributions `test1.dat, test2.dat, test3.dat`. From this, give your estimates of the statistical error of $\overline{X}$ and of the autocorrelation time. See slides for Practice session 2. Below are the results for the first few values of m for `test1.dat`, so you can check if your binning code is correct:

m	1	2	4	8	16
$s_{\overline{X}^{(m)}}$	0.00225	0.00313	0.00432	0.00585	0.00765

- If you have them, the plots and estimates required in task n.4. Please put the estimates in the text of the email.

Final Submission (December 23):

- A brief pdf document containing the plots, estimates, and intepretations required in tasks n.4, n.6, n.7, n.8, plus any optional task you may have done. For the required part there should be no more than two pages written text in total, not counting the figures.

  Do **not** include a long introduction or detailed explanation of the methods, unless it is to justify why you did something different than what was requested. **Important:** do not use png, jpeg, or similar formats for the figures. In scientific publications one should produce figures in eps (Encapsulated Postscript) or pdf, which can be zoomed in without losing quality.

- The Open Office document `results2DIsing.ods` with your data, one line for each pair (L, T), including also $L = 20$. Include also any optional observables you have measured.

- All your codes (human-readable and with comments)

- Input files and random seeds used, and all is needed to reproduce the results.

- A README of no more than 20 lines in text format telling me how to reproduce your results: how to compile the code, how to run the code, how to generate the plots, etc.

- Please *do not* include large files with long time series.

Grading of the computational project: 10% submission practice 1, 10% submission practice 2, 80% final submission.