Project 4

Contents

> **Note**
>
> While working on the project, check this page regularly in case of small updates (typo fixes, added hints, etc.)

Practicalities

- **Deadline**: Wednesday, November 20, 23:59.
- **Format:**
  - A scientific report, typeset in LaTeX, delivered as a pdf file on Canvas.
  - Use the report template we have provided here.
  - Code (with comments, of course) on a UiO GitHub repo (github.uio.no), with the URL to you repo written in the pdf document.
  - You *must* deliver via your group on Canvas (even if you are working alone).
- **Collaboration:** We *strongly* encourage you to collaborate with others, in groups of up to three students. The group hands in a single pdf. Remember to list everyone's name in the pdf.
- **Reproducibility:** Your code should be available on a GitHub repo. You can refer to relevant parts of your code in your answers. Make sure to include a README file in the repo that briefly explains how the code is organised, and how it should be compiled and run in order for others to reproduce your results.
- **Figures:** Figures included in your LaTeX document should preferrably be made as vector graphics (e.g. `.pdf` files), rather than raster graphics (e.g. `.png` files).

Introduction

The topic of this project is the **Ising model** in two dimensions. We will use this simple model to explore temperature-dependent behaviour in ferromagnets. A particular goal is to numerically estimate the **critical temperature** at which our 2D system undergoes a **phase transition**, from a magnetised phase to a phase with no net magnetisation.

A basic introduction to the Ising model will be given in the lectures. For a more detailed introduction, see Chapter 13 (in particular 13.3 and 13.4) in Morten Hjorth-Jensen's lecture notes.

On the methodological side, our focus in this project is

- the Markov Chain Monte Carlo method for sampling from probability distributions of many variables, and how the resulting samples can be used to approximate probability distributions and expectation values for derived quantities; and
- using parallelisation to speed up our code.

Definitions, notation and units

We will study 2D square lattices of spins s_i. Below we summarise our notation and the main definitions we will need:

- Each individual spin s_i has only two possible states: $s_i = -1$ and $s_i = +1$.
- A lattice of length L contains N spins in total, i.e. $N = L^2$.
- A **spin configuration** (or **microstate**) refers to the spin state of the *entire system*. Here we denote this using vector notation
$$\mathbf{s} = (s_1, s_2, \ldots, s_N),$$
but in your code you probably want to use an $L \times L$ matrix to represent the spin configuration.
- The total energy of the system is given by
$$E(\mathbf{s}) = -J \sum_{\langle kl \rangle}^{N} s_k s_l$$
Here $\langle kl \rangle$ means that the sum goes over all *neighbouring pairs* of spins, without double-counting, and J is the **coupling constant**. We see that in this model (with no external magnetic field) the total system energy is fully determined by the interaction between neighbouring spin pairs, and J is simply a constant that sets the energy associated with each such interaction.
- We will use *periodic boundary conditions* for our lattice. Thus, all spins will have exactly *four neighbours* (left, right, up, down).
- The total magnetisation of the system is simply given by the sum over all the spins in the system:
$$M(\mathbf{s}) = \sum_{i}^{N} s_i$$
- To ease the comparison of results for different lattice sizes, we will mostly use the *energy per spin* (ϵ) and *magnetisation per spin* (m):
$$\epsilon(\mathbf{s}) = \frac{E(\mathbf{s})}{N}$$
$$m(\mathbf{s}) = \frac{M(\mathbf{s})}{N}$$
- Given a system temperature T, the probability for the system state $\mathbf{s}$ is given by the *Boltzmann distribution*:
$$p(\mathbf{s}; T) = \frac{1}{Z} e^{-\beta E(\mathbf{s})}$$
Here the normalisation constant Z is the so-called **partition function** (see below), and β is the "inverse temperature":
$$\beta = \frac{1}{k_B T}$$
with k_B being the Boltzmann constant.
- The partition function Z is given by
$$Z = \sum_{\text{all possible } \mathbf{s}} e^{-\beta E(\mathbf{s})}$$
- **Units**: We are working with unitless spins, which means that the coupling constant J has units of energy. Thus we can take J as the base unit for energy, and use it together with the Boltzmann constant k_B to formulate all other units that we'll need. So our units are
$$[E] = J$$
$$[T] = J/k_B$$
$$\ldots$$
(You can figure out the rest.)

Problems

Problem 1

Assume a 2×2 lattice with periodic boundary conditions.

- **a)** Summarise all possible states of the system by setting up a table with four coloumns showing the following quantities
  - number of spins in state +1
  - total energy
  - total magnetisation
  - degeneracy
- **b)** Find analytical expressions for
  - Z
  - $\langle \epsilon \rangle$
  - $\langle \epsilon^2 \rangle$
  - $\langle |m| \rangle$
  - $\langle m^2 \rangle$
  - the heat capacity, normalised to number of spins:

$$\frac{C_V(T)}{N} = \frac{1}{N}\frac{1}{k_B T^2}\mathrm{Var}(E) = \frac{1}{N}\frac{1}{k_B T^2}\left[\langle E^2 \rangle - \langle E \rangle^2\right] = N\frac{1}{k_B T^2}\left[\langle \epsilon^2 \rangle - \langle \epsilon \rangle^2\right.$$

  - the susceptibility, normalised to number of spins:

$$\frac{\chi(T)}{N} = \frac{1}{N}\frac{1}{k_B T}\mathrm{Var}(M) = \frac{1}{N}\frac{1}{k_B T}\left[\langle M^2 \rangle - \langle |M| \rangle^2\right] = N\frac{1}{k_B T}\left[\langle m^2 \rangle - \langle |m| \rangle\right.$$

We will use these analytical results to test our code.

> **Note**
>
> Above we have replaced $\mathrm{Var}(M) = \langle M^2 \rangle - \langle M \rangle^2$ with $\langle M^2 \rangle - \langle |M| \rangle^2$. The replacement $M \to |M|$ is a trick often used in Monte Carlo studies of this 2D Ising model, and we will also use this in the numerical tasks below. So to aid our later comparison between analytical and numerical results, we make the replacement $M \to |M|$ also in the analytical expression above.
>
> *More details:* The $M \to |M|$ replacement makes it easier to compare and combine results from different runs (e.g. repeated runs at the same or similar temperatures), because we don't have to worry about whether the system randomly ended up in a state with mostly -1 spins or a state with mostly $+1$ spins. We simply focus on the question "*How strong* magnetisation do we expect to observe at temperature T?", rather than the question "*What* magnetisation do we expect to observe at temperature T?".
>
> But it's worth keeping in mind that, with no external magnetic field to pick out a preferred direction, the expectation value $\langle M \rangle$ really is 0 for all temperatures – that is, even at very low temperatures, where there should be overwhelmingly high probability to observe the system in an ordered state, i.e. a state with $|M| > 0$.
>
> The reason for the expected value $\langle M \rangle = 0$ even at low temperatures is that for each state with some positive magnetisation M, there exists a "mirror state" with *all spins flipped*, which means the magnetisation is $-|M|$, and this state has the same probability under the Boltzmann distribution as the original state. So the expected value $\langle M \rangle = 0$ is analogous to how the expected value for the amount of money earned on fair bets with 50/50 coin flips is zero, even though zero is an impossible outcome for individual trails – every trail gives either a win or a loss.

Problem 2

In our Monte Carlo code we will repeatedly need the Boltzmann factor $e^{-\beta \Delta E}$, where

$$\Delta E = E_{\text{after}} - E_{\text{before}}$$

is the energy shift due to *flipping a single spin*.

a) Consider a 2D lattice of arbitrary size ($L > 2$). Show that there are only five possible values ΔE can take.

b) How can you use this result to avoid repeatedly calling the exponential function `exp(...)` in your code?

Problem 3

There are many ways to implement periodic boundary conditions. The simplest is a set of if-tests. That works, and may be a useful starting point, but if-tests inside loops can in some cases slow down your code. (It can hinder compiler optimisations.) Can you come up with a more efficient way of implementing periodic boundary conditions in your code?

Problem 4

Time to actually write some code!

a) Implement a code for the Ising model which uses the Markov Chain Monte Carlo approach to sample spin configurations $\mathbf{s}$ and computes

- $\langle \epsilon \rangle$
- $\langle |m| \rangle$
- C_V/N
- χ/N

b) Validation: For a temperature of $T = 1.0\, J/k_B$, compare your results to the analytical results from Problem 1. (Feel free to compare for other temperature values as well, or simply make plots of these quantities versus temperature.)

c) How many **Monte Carlo cycles** do you need to get good agreement with the analytical result? (Here *one Monte Carlo cycle* corresponds to N *attempted spin flips*.)

Problem 5

Now we will use a lattice with size $L = 20$ and study the **burn-in time** (or **equilibration time**), measured in number of Monte Carlo cycles.

a) Make a plot that shows the current state energy ϵ (not $\langle \epsilon \rangle$) versus the number of Monte Carlo cycles done. The plot should contain four graphs:

- $T = 1.0\, J/k_B$, starting from *ordered* initial state (all spins pointing the same way)
- $T = 1.0\, J/k_B$, starting from *unordered* (random) initial state
- $T = 2.4\, J/k_B$, starting from *ordered* initial state (all spins pointing the same way)
- $T = 2.4\, J/k_B$, starting from *unordered* (random) initial state

Make a similar plot for the magnetisation $|m|$ of the current state versus the number of Monte Carlo cycles done.

- It may be interesting to also include plots showing how the numerical estimates of $\langle \epsilon \rangle$ and $\langle |m| \rangle$ evolve with the number of Monte Carlo cycles included in the estimation

b) Based on these plots, how long would you say the burn-in time is?

Problem 6

Now let's move beyond mere expectation values and see how we can use our MCMC to estimate the full probability distribution for ϵ.

For $L = 20$, approximate the probability function $p_\epsilon(\epsilon; T)$ for $T = 1.0\, J/k_B$ and $T = 2.4\, J/k_B$ by creating normalised histograms of generated ϵ samples. (Keep an eye on bin widths and remember that ϵ is not a continuous variable.) Comment on your results, in particular on the variance of the two distributions.

Problem 7

Now parallelise your code using either OpenMP or MPI. Perform some timing tests to estimate the speed-up factor resulting from the parallelisation.

> **Note**
>
> See our updated Introduction to OpenMP page for instructions on how to build your code with OpenMP enabled.

> **Note**
>
> There are basically three different approaches you can take here, corresponding to three different levels at which you can parallelise your code:
>
> 1. *At the temperature level:* You can parallelise the outer loop over temperature values (see Problem 8, which is where parallelisation becomes important)
> 2. *At the level of MCMC "walkers":* You can use multiple threads to perform multiple independent MCMC runs in parallel and combine their results.
> 3. *At the level of spin flipping:* You could parallelise the part of the code where you generate new states by attempting to flip N spins.
>
> We recommend you do either approach 1 or 2. (You only need to do one approach.) Approach 1 is probably the easiest. Approach 2 is the standard approach to parallelising a single MCMC run (i.e. when there isn't any scan over some external parameter, like T in our case.) Approach 3 would by far be the most complicated and it is probably also the least efficient.

Problem 8

Now we will start investigating phase transitions.

> **Note**
>
> For a brief introduction to the phase transition of the 2D Ising model, see Chapters 13.4 and 13.4.1 of Morten Hjorth-Jensen's lecture notes. See also the brief theoretical summary for Problem 9.

- **a)** For lattices of size $L = 40, 60, 80, 100$ (and higher if you want), make plots of $\langle\epsilon\rangle$, $\langle|m|\rangle$, C_V/N and χ/N as function of temperature T. (Plot the results for the different L values in the same figure.) Focus on temperatures in a range around $T \in [2.1, 2.4]\, J/k_B$. Make sure to use sufficiently small temperature steps, at least in the region where the graphs change most rapidly.
- **b)** Do you see an indication of a phase transition?
- **c)** What is the critical temperature $T_c(L)$ for the different lattice sizes L? Use the χ/N or C_V/N results to determine $T_c(L)$.

Note

Some suggestions in case you are short on time because the code is slower than expected:

- Make sure you're not running way more temperature points than necessary. A useful strategy: First run a scan with large T steps, to roughly identify the subregion of T values where the graphs change most quickly. (E.g. only 10 points on the suggested T range can be sufficient here.) Then run a second scan that zooms in on the much smaller T range needed to identify T_c. Again, you can probably get good results with as few as 10 T points – though more points always make for better results.
- If you're having trouble with OpenMP/MPI parallelisation, remember that to just make sure you get the numerical results you need, you can always do "dummy parallelisation" by just running multiple instances of your program. (Use separete terminal windows, or run your program in the background with the `&` command, e.g. `./main.exe <command line options> &`. You can view your background jobs with `jobs` and bring them to the foreground with `fg <job number>`.) This "manual" way of splitting up tasks works well assuming that your program can take options like `L`, `Tmin`, `Tmax`, `deltaT` (or `n_Tsteps`) as command-line arguments.
- Perhaps you can afford to reduce the number of Monte Carlo cycles a bit and still get decent results? E.g. halving the number of cycles should halve the program runtime.
- Keep in mind that you don't need to finish all the runs for Problem 8 to get started on Problem 9. To get a first result for Problem 9 you basically only need to identify T_c for two of the L values in Problem 8. So you can do Problem 9 in parallel with the runs for Problem 8, and then just update your plots/numbers as new runs in Problem 8 finishes.

Problem 9

The critical temperature for an *infinite* ($L = \infty$) 2D Ising model was found analytically by Lars Onsager in 1944. The result is

$$T_c(L = \infty) = \frac{2}{\ln(1 + \sqrt{2})} J/k_B \approx 2.269\, J/k_B$$

Our task now is to try to numerically estimate $T_c(L = \infty)$ using our numerical results for $T_c(L)$ of *finite* systems.

Using your set of L and $T_c(L)$ values from Problem 8 as data points, and the scaling relation

$$T_c(L) - T_c(L = \infty) = aL^{-1} \tag{22}$$

where a is a constant, extract an estimate for $T_c(L = \infty)$ and compare it to Onsager's analytical result.

Note

Hint: If you are unsure about how to approach this, try to first plot your data points in such a way that they, according to the relation above, should fall on a straight line.

Note

Hint 2: If you find that doing a linear fit (a.k.a. linear least squares, a.k.a. linear regression, …) could be useful, we don't expect you to discuss that method in detail in your report. Feel free to simply apply one of the many tools available, e.g. `scipy.stats.linregress`.

Theoretical background: *What follows here is a very brief summary. For more information see Chapters 13.4 and 13.4.1 of Morten Hjorth-Jensen's lecture notes, and references therein.*

One of the many fascinating aspects of so-called critical phenomena is that vastly different physical systems can exhibit the same behaviour close to their critical point (here: critical temperature). This behaviour is described by power laws, with exponents called *critical exponents*. For the *infinite* 2D Ising model, the mean magnetisation, heat capacity and susceptibility behave as

$$\begin{aligned}\langle |m| \rangle &\propto |T - T_c(L=\infty)|^{\beta} \\ \frac{C_V}{N} &\propto |T - T_c(L=\infty)|^{-\alpha} \\ \frac{\chi}{N} &\propto |T - T_c(L=\infty)|^{-\gamma}\end{aligned}$$

for temperatures T close to T_c, with critical exponents $\beta = 1/8$, $\alpha = 0$ and $\gamma = 7/4$.

> **Note**
>
> A critical exponent of 0 is just the power-law way of expressing *logarithmic* behaviour.

Both C_V/N and χ/N diverge near T_c. Similarly, the *correlation length* (here: the typical lenght scale of regions with aligned spins) ξ also diverges near the critical point,

$$\xi \propto |T - T_c(L=\infty)|^{-\nu} \tag{23}$$

with critical exponent $\nu = 1$. However, since we are studying a *finite* system, the largest correlation length is $\xi = L$, which we associate with the critical temperature of the finite system, $T = T_c(L)$. Inserting this into equation (23) gives rise to equation (22) above.

Using this relation we see that, when T is close to $T_c(L)$, we expect the mean magnetisation, heat capacity and susceptibility to depend on the system size L as

$$\begin{aligned}\langle |m| \rangle &\propto L^{-\beta/\nu} \\ \frac{C_V}{N} &\propto L^{\alpha/\nu} \\ \frac{\chi}{N} &\propto L^{\gamma/\nu}.\end{aligned}$$

Good luck!

Code snippets

- See the code snippets in **`code_examples/omp_parallelization`** for some simple examples on parallelisation using OpenMP.
- For some illustrations on how you can use the C++ `<random>` library for random number generation, see code examples in **`code_examples/random_number_generation`**. These examples use a Mersenne Twister generator. Some of the examples demonstrate how one can do seeding when working with multiple OpenMP threads.

By the FYS3150 teaching team