Project 2

Here you study the thermodynamic properties of the two-dimensional Ising model. It consists of a system of interacting spins on a lattice and serves as a simple model for ferromagnetism. It allows you to study the behavior of a system around a critical point. More specifically, the spins sit on a square lattice of size $N \times N$. s_i is the value of the spin at site i and it can take the values $+1$ and -1 as it either points "up" or "down". The Hamiltonian is given by

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} s_i s_j - H \sum_i s_i.$$

The first term accounts for a coupling between spins. The summation is over nearest neighbor pairs on the lattice (each spin has four nearest neighbors; we assume periodic boundary conditions). J is a coupling constant. We assume here $J > 0$ which means that the spins prefer to be aligned to lower the energy of the system; for simplicity, set $J = 1$. The second term in the Hamiltonian accounts for the presence of an external magnetic field H which favors the spins to have the same sign as that of the field. In the following we do not consider this term, i.e. we always set $H = 0$. You will find that—below a critical temperature—the system shows overall alignment of the spins, even though an external magnetic field is absent, i.e. the system exhibits ferromagnetic behavior.

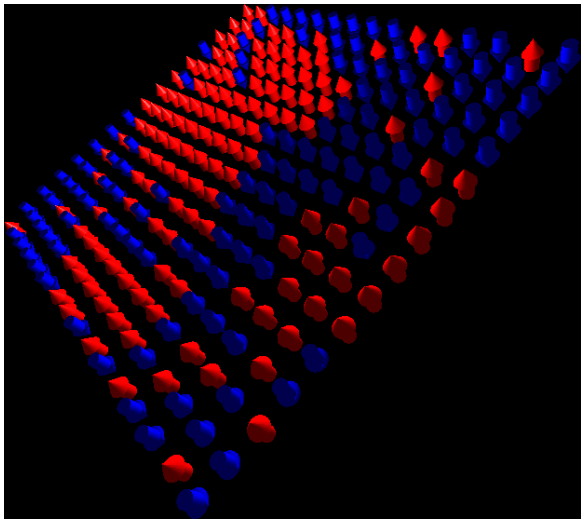

Figure 1: A 2D-array of "up" and "down" spins

Monte Carlo Simulation

If you just want to study the properties of a system at thermal equilibrium but are not interested in its dynamics, the Monte Carlo method is a very powerful approach. As you know, statistical mechanics typically requires the calculation of high-dimensional integrals. For instance, the average energy of a system follows from integrating the energy (weighted by the Boltzmann weight) over all degrees of freedom. A natural way to arrive at the Monte Carlo method is by starting with the question of how to calculate such high-dimensional integrals numerically.

We start first with a one-dimensional integral, $\int_a^b f(x)\,dx$. If we do not know the exact solution of this integral, we could try to determine its value numerically as follows:

$$\int_a^b f(x)\,dx = (b-a)\int_a^b \frac{1}{b-a} f(x)\,dx \tag{1}$$

$$= (b-a)\int_a^b p(x) f(x)\,dx \tag{2}$$

$$\approx \frac{b-a}{N}\sum_{i=1}^{N} f_i \tag{3}$$

with $f_i = f(x_i)$. Here the x_i's are drawn randomly from the uniform distribution in the interval $[a,b]$. The scheme is called Monte Carlo integration. To understand how to arrive at the sum on the right hand side, we inserted in the middle of Eq. 3 an extra step, a trivial rewriting of the original integral. This expression can be interpreted as calculating the *expectation value* of the function of $f(x)$ for the probability distribution $p(x) = 1/(b-a)$. The expression on the right hand side of Eq. 3 is then an estimation of this integral by drawing random numbers x_i from the probability distribution $p(x)$.

How accurate is this method? Let us calculate the variance:

$$\sigma^2 = \left\langle \left(\frac{b-a}{N}\sum_{i=1}^{N} f_i \right)^2 \right\rangle - \left(\left\langle \frac{b-a}{N}\sum_{i=1}^{N} f_i \right\rangle \right)^2 \tag{4}$$

where the angular brackets denote the average over all possible realisations of sequences of N random coordinates x_i. Let us denote the average of f on the interval $[a,b]$ by $\overline{f}$. Using $\langle f_i \rangle = \overline{f}$, $\langle f_i^2 \rangle = \overline{f^2}$ and, for $i \neq j$, $\langle f_i f_j \rangle = \langle f_i \rangle \langle f_j \rangle = \overline{f}^2$ we can simplify Eq. 4 to

$$\sigma^2 = \frac{(b-a)^2}{N}\left(\overline{f^2} - \overline{f}^2 \right). \tag{5}$$

Therefore the standard deviation σ from the real value, i.e. the error, scales as $1/\sqrt{N}$. *This is really inefficient!*

There are much better methods available to determine $\int_a^b f(x)\,dx$ numerically with errors that scale as $1/N^k$ with k being a positive integer. The most simple method leads to $k = 1$ and goes as follows: Define an equidistant grid on the interval $[a, b]$. The grid points are then given by $x_i = a + ih$ with $h = (b - a)/N$ and $i = 0, ..., N$. The integral is then approximated by

$$\int_a^b f(x)\,dx \approx h \sum_{i=0}^{N-1} f_i, \tag{6}$$

see also Fig. 2. This is presumably what you would have done if asked to calculate an integral numerically (and rightly so!). It matches the "natural" definition of integrals.

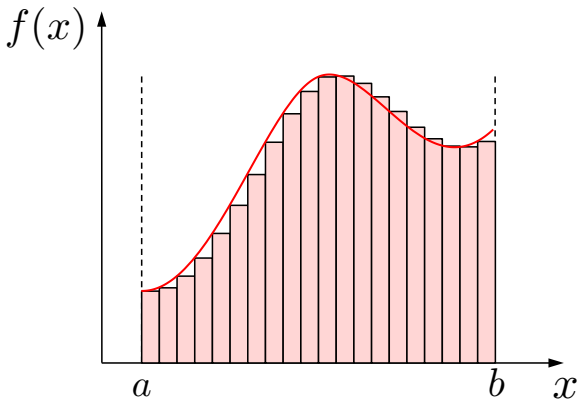

Figure 2: A standard numerical method for calculating $\int_a^b f(x)\,dx$ involves dividing the interval $[a, b]$ into an equidistant grid with corresponding N values x_i. This particular example (Eq. 6) produces an error that scales like $1/N$. It is straightforward to design approximations that lead to smaller errors.

Let us estimate the error. Assume for simplicity that $f(x)$ is a continuous function on the interval $[a, b]$ which has a finite slope everywhere with an absolute value that is smaller or equal to a value M. Then for each subinterval our error of our estimate cannot exceed $Mh/2$ (this maximal deviation from the estimated value occurs for the special cases $f(x) = Mx$ and $f(x) = -Mx$). The overall error follows from Eq. 6 to be smaller than

$h \times NMh/2 = (b-a)\,Mh/2$. Therefore the error scales as $1/N$. This clearly outperforms Monte Carlo integration which only scales as $1/\sqrt{N}$. Even better methods exist by using trapezoids rather than rectangles or by finding a clever way of distributing the points on the interval.

Why would one then ever consider to numerically determine integrals using the Monte Carlo integration scheme? As we demonstrate now this method is very powerful and outperforms other methods when it comes to calculating high-dimensional integrals. Suppose one wants to calculate the integral over a function f that is defined on a d-dimensional hypercube L^d. We can estimate the integral by evaluating f on an equidistant grid by dividing the hypercube into N little hypercubes, each of volume $h^d = L^d/N$. The standard deviation is then proportional to h^k (with some $k \geq 1$, e.g. $k = 1$ as shown in the example above) which scales here as $1/N^{k/d}$. Therefore these numerical methods do not work very well, as an increase in N hardly improves their performance. On the other hand, the Monte Carlo integration is not affected by the dimensionality of the system. You can check this by inspecting again the argument that led to Eq. 5. The standard deviation always scales as $1/\sqrt{N}$.

Now, the type of integral we need to estimate in statistical physics (and hence to study our lattice of spins) is typically of the form

$$\langle A \rangle = \frac{1}{Z} \int A(x)\, e^{-\beta H(x)} dx \tag{7}$$

where $\beta = 1/k_B T$ and x runs over all the possible states e.g. the positions and momenta of all the particles (for example, A could be the energy). Since this is a very high-dimensional integral, Monte Carlo integration is clearly the method of choice.

However, there is an additional complication. The function $A(x)\, e^{-\beta H(x)}$ is typically very sharply peaked in phase space, see Fig. 3 and in most of phase space this function is practically zero. It is thus very inefficient to perform a uniform random sampling as most of the x_i's happen to be in regions in phase space that do not contribute to the integral in any significant way. Despite this complication, the error still scales as $1/\sqrt{N}$; however, the prefactor is huge.

To solve this problem one introduces importance sampling. Suppose the probability distribution on phase space is given by $p(x)$, with $p(x) \geq 0$ everywhere, a function normalized to one and possibly sharply peaked. Then

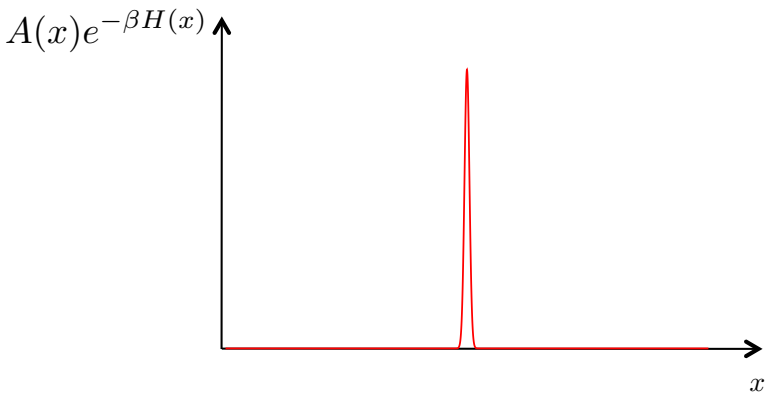

Figure 3: In statistical mechanics one typically has to compute very high-dimensional integrals. This figure is therefore misleading as it only shows one-dimensional integration. Another typical feature, however, is displayed properly, namely that the functions one needs to integrate, e.g. $A\left(x\right)e^{-\beta H(x)}$, are typically very sharply peaked.

the idea is to evaluate the integral $\int p\left(x\right)A\left(x\right)dx$ as follows:

$$\int A\left(x\right)p\left(x\right)dx \approx \frac{1}{N}\sum_{i=1}^{N}A\left(x_i\right) \tag{8}$$

where x_i is sampled according to the now non-uniform probability distribution $p\left(x\right)$. How can this be achieved in practice? The answer leads to the Metropolis algorithm that lies at the heart of Monte Carlo simulations.

Markov chains and the Metropolis algorithm

We want to find a way for the system to "naturally" get towards the distribution $p(x)$ that is good for our system. We can then use it to navigate around and sample the integral.

Let's start in very general terms. We introduce a so-called *Markov chain*, a set of random states $\{x_i\} = \{x_1, x_2, x_3, ...\}$ where the probability of x_{i+1} depends on x_i only (i.e. not on any of the previous ones; there is no memory of the past history of the system). This can be described by transition probabilities $T\left(x \to x'\right)$ between states x and x'. These fulfill

$$\sum_{x'} T\left(x \to x'\right) = 1, \tag{9}$$

reflecting the fact that you have to go *somewhere* from x. We can now write a so-called master equation for the probabilities $p(x)$ of each state at "time" (i.e. step) i:

$$p\left(x, i+1\right) = p\left(x, i\right) - \sum_{x'} p\left(x, i\right) T\left(x \to x'\right) + \sum_{x'} p\left(x', i\right) T\left(x' \to x\right). \quad (10)$$

The interpretation of this equation is that at the next time step, the probability to be in state x is reduced by transitions to other states (2nd term on the right hand side) and increased by transition from other states to state x (last term on right hand side).

We want to sample the system at thermal equilibrium, i.e. we are interested in a stationary probability distribution, $p\left(x, i+1\right) = p\left(x, i\right) = p\left(x\right)$. In this case Eq. 10 simplifies to

$$\sum_{x'} p\left(x, i\right) T\left(x \to x'\right) = \sum_{x'} p\left(x', i\right) T\left(x' \to x\right). \quad (11)$$

Thinking back about the integral we wanted to compute, the values of $p\left(x\right)$ are known (Boltzmann weight) but we still do not know the transfer probabilities $T\left(x \to x'\right)$. As this relation is hard to solve, one typically uses stronger constraints, namely requiring

$$p\left(x, i\right) T\left(x \to x'\right) = p\left(x', i\right) T\left(x' \to x\right). \quad (12)$$

This is called *detailed balance* and it is a special case that fulfills Eq. 11.

To proceed further we separate $T\left(x \to x'\right)$ into two factors:

$$T\left(x \to x'\right) = w_{xx'} \times A_{xx'} \quad (13)$$

The first factor, $w_{xx'}$, denotes the trial step probability: the probability to propose a new state x' given a state x. The second factor, $A_{xx'}$, is the acceptance probability: the probability to accept the proposed new state x' if the system was in state x before.

The trial step probabilities should clearly be symmetric, $w_{xx'} = w_{x'x}$. One should always check whether this is indeed true when setting up a Monte Carlo simulation.

With this decomposition, the detailed balance condition, Eq. 12, simplifies to

$$\frac{A_{xx'}}{A_{x'x}} = \frac{p\left(x'\right)}{p\left(x\right)}. \quad (14)$$

This relation can be satisfied as follows:

$$A_{xx'} = \begin{cases} 1 & \text{if } p(x') > p(x) \\ p(x')/p(x) & \text{if } p(x') < p(x). \end{cases} \tag{15}$$

Now we have all information to present the steps to be taken in the Metropolis algorithm, namely

1. Start with a random initial state x_i (with $i = 0$). In our case, that is a choice of up-and-down spins.

2. Generate a state x' from x_i (such that $w_{x_i x'} = w_{x' x_i}$). In our case that is a configuration with a different choice of up-and-down spins.

3. If $p(x') > p(x_i)$, $A_{xx'} = 1$, i.e. set $x_{i+1} = x'$. Otherwise, accept move $(x_{i+1} = x')$ with probability $q = p(x')/p(x_i)$ or reject move $(x_{i+1} = x_i)$ with probability $1 - q$.

4. Continue with (2)

In the context of statistical physics, $p(x)$ is the Boltzmann weight, i.e. $p(x) \propto e^{-\beta H(x)}$. As the probability decreases monotonously with H one can reformulate step (3) as follows. Instead of using probabilities one uses energies, i.e. instead of checking $p(x') > p(x_i)$ one checks $H(x') < H(x_i)$. In other words, if the trial move lowers the energy it is always accepted, and if it increases the energy by an amount ΔE, it is accepted with a probability $e^{-\beta \Delta E}$. This has a clear physical interpretation. The system will try to move to lower total energy. The higher the temperature, the higher the probability to flip a spin and go to a state with more energy.

Phase transition in the two-dimensional Ising model

There exists obviously no Molecular Dynamics simulation of the Ising model. Instead it is an ideal system to apply and understand the working of the Monte Carlo approach. In order to set up the Metropolis algorithm we must choose specific values for the trial step probabilities $\omega_{xx'}$, see Eq. 13. As we are on an $N \times N$ lattice, this is most naturally done as follows:

$$\omega_{xx'} = \begin{cases} 1/N^2 & \text{if } x \text{ and } x' \text{differ by one spin} \\ 0 & \text{otherwise.} \end{cases}$$

This means the following. Suppose your N^2 spins are in state x. You create now a trial state x' by picking one spin *at random* and flip it. You then

calculate the energy difference $\Delta E = \mathcal{H}(x') - \mathcal{H}(x)$. To do this calculation you only have to calculate the change in the interaction energy of the picked spin with its four neighbors. If $\Delta E < 0$ you always accept the move and if $\Delta E > 0$ you accept it only with a probability $e^{-\beta \Delta E}$.

One complication is that you need to account for the periodic boundary conditions. As in the previous exercise, this can be most easily dealt with using the Modulo operator. Suppose you calculate the interaction of a spin at `[nx, ny]` with the neighbour to the right at `[nx + 1, ny]`. This could cause a problem if you sit at the right boundary of the lattice where you should look up the spin on the left boundary of the lattice instead. You can automatically account for this if you look up the spin at `[(nx + 1) % L, ny]` .

It is useful to always keep track of the total energy of the system. To do so calculate this energy at the beginning of the simulation by summing about all interactions between the spins. Then, as you run the simulation, simply add ΔE to the total energy whenever there is a successful spin flip. Similarly, you can always keep track of the total magnetization, $M = \sum_i s_i$. Moreover, since $e^{-\beta \Delta E}$ can take only five different values, you can calculate these values only once for a given temperature and store them in an array (in fact, you actually need only two values, namely for the cases where $\Delta > 0$).

First run some simulations at various fixed temperatures at zero external magnetic field. Plot the magnetization per spin, $m = M/N^2$, as a function of "time", measured here in Monte Carlo steps per lattice site (also called one sweep of the lattice). The reason why we measure time this way is that after each sweep each spin had on average a chance to attempt one flip (independent of the total size of the lattice). You can start either with a random spin configurations (corresponding to an infinite temperature) or with a state where all spins point at the same direction (corresponding to a state at zero temperature). A reasonable size of the lattice is 50×50, i.e. $N^2 = 2500$ spins but it can also useful to study smaller or larger lattices. It is also recommended to plot the spin configurations at the end of your simulation (e.g. Fig. 1).

There is a critical temperature below which there is a spontaneous magnetization. For the two-dimensional Ising model the critical temperature is known analytically (for an infinite lattice):

$$\frac{k_B T_c}{J} = \frac{2}{\ln\left(1 + \sqrt{2}\right)} \approx 2.269. \tag{16}$$

First Milestone

Code up and play around with this system. Try maybe $N = 10$ first then $N = 20$ and eventually move to $N = 50$.

You should run several simulations for various temperatures, (at least) from $T = 1.0$ to $T = 4.0$ (set $J = k_B = 1$) in steps of 0.2. For each temperature you need to get a rough estimate of the equilibration time by inspecting curves of the magnetization as a function of time.

It is best to perform two simulations starting at different (or even the same) initial conditions and to see when the two curves reach similar values in magnetization.

Hint: You might observe that the time to equilibration is larger close to the critical temperature.