Final Project Report
Ising Model with Cellular Automata

Frausin Enrico*

February 2024

Problem

a In the cellular automata implementation of the Ising model, the energy of the entire lattice is fixed, and a spin may flip only if the energy of the lattice does not change. Such a situation occurs if a spin has two up neighbors and two down neighbors. Hence the update rule is to flip the spin at i, that is, $s_i \to -s_i$, if it has precisely two up neighbors; otherwise do not change si. But because we want to update all sites simultaneously, we have a problem. That is, if two neighboring spins have opposite signs and each has a total of two up neighbors, then flipping both spins would change the total energy. Why? The trick is to divide the lattice into two kinds of sites corresponding to the red and black squares of a checkerboard. First we update simultaneously all the red squares (hence keeping their neighbors, the black squares fixed), and then we update simultaneously all the black squares. Implement this cellular automata update of the Ising model by modifying Program ca2 and write a subroutine to compute the mean magnetization per spin and the total energy per spin E/N. (The latter should not change with time.) The average is over the different configurations generated by the cellular automata updates. Use periodic boundary conditions.

b Compute the mean magnetization as a function of E/N for a square lattice with L= 20. One way to generate an initial configuration is to let each spin be up with probability p. Allow the system to "equilibrate" before accumulating values of the magnetization. Does the

*ENRICO.FRAUSIN@studenti.units.it

mean magnetization change from being close to zero to being close to unity as E/N is lowered? If such a qualitative change occurs, we say that there is a phase transition. To improve your results, average over many different initial configurations with the same value of E/N and determine the dependence of the mean magnetization on E/N. Because the same value of p will occasionally lead to different initial energies, write a subroutine that flips spins until the desired energy is obtained.

c Repeat part (b) for a 40×40 lattice. Do your qualitative conclusions change?

d One difficulty with the cellular automata version of the Ising model is that it is not ergodic, that is, there are configurations with the same energy that cannot be reached for a given initial condition. In Chapter ?? we will see how to avoid this problem using Monte Carlo methods. Here we might allow the total energy E0 to vary a little, say $\pm$nJ , where n is an integer. During a run we can periodically flip a spin at random such that the total energy is in the range E0 $\pm$nJ. Try this method for different values of n. Do your results for the mean magnetization change significantly?

1 Code Structure

The project is structured as follows: in the notebook `Cell_ising.ipynb`, the two main classes of the program are initially defined: the `Spin_matrix()` class and the `Model()` class. The first collects the instructions to initialize and update a spin lattice using cellular automata. It is also possible to calculate and modify the energy and magnetization values of the lattice, and optionally visualize the spin configuration graphically. The second class handles the management of simulation data collection, namely the generation of energy-magnetization pairs, primarily through the `Model.driver()` function. This function requires mandatory input of the number of steps for the evolution of each individual lattice. By optionally inserting the appropriate parameters, one can control: the number of energy-magnetization data pairs desired, the number of equilibration steps, the possibility of initializing lattices at exact energies, the number of iterations per single energy over which to average, the magnetization of individual lattices (positive or negative), and the possibility of varying the energy of individual lattices to compensate for the non-ergodicity of temporal evolution. `Model()` also allows saving data to a file and displaying energy-magnetization graphs.

The evolution via cellular automata of the system has been completely implemented in the dedicated Fortran module `ising_auto.f90`. Within it are the subroutines and appropriate functions to calculate the energy and magnetization of a spin lattice, globally evolve the cellular automata by one time step, change some spins of the lattice until a given energy is reached, and vary the energy of a lattice within a determined interval. It is also possible to control the seed of the random number generator through the subroutines `seed_sizer()` and `set_rng_seed`, for reproducibility of simulations.

The seed used before generating each graph that will be presented throughout the report is: [210, 3074, 383, 267, 4392, 5436, 4686, 9494]. Without loss of generality, the coupling constant J of the energy has been assumed equal to 1.

2 Point (a)

Point a) consists of writing the code necessary for the temporal evolution of spin lattices.

In a simulation with cellular automata of the Ising model, the energy is kept fixed at a determined value. To do this, a spin is flipped if and only if it has exactly two 'up' nearest neighbors and two 'down' nearest neighbors, since the energy change in this case is zero, given by

$$\Delta E = 2s_{i,j}(s_{\text{up}} + s_{\text{down}} + s_{\text{right}} + s_{\text{left}}) \tag{1}$$

where $s_{i,j}$ indicates the spin at position ij in the lattice and s_{up}, s_{down}, s_{right}, s_{left} indicate the nearest neighbors of $s_{i,j}$ with periodic boundary conditions.

3 Point (b)

3.1 Energy per Spin-Probability Fit

A problem in generating lattices is their initialization with a determined value of energy per spin E/N. The chosen method was the one suggested by the problem text, namely each site at the moment of spin generation is set to -1 with probability p. The problem remains of determining the dependence relationship of the energy per spin of the lattice on the initialization probability. Hypothesizing a correct fit function is fundamental, since the system dynamics does not admit energy variations. An incorrect fit function would generate energy values too concentrated in some intervals and too dispersed in others, thus preventing a homogeneous analysis of the energy

spectrum of the system. Through a brief script, it was seen that the optimal fit function for the energy dependence on probability is quadratic, namely $E/N = ap^2 + bp + c$, where the values of a, b, and c found are: $a = -8$, $b = +8$, $c = -2$.

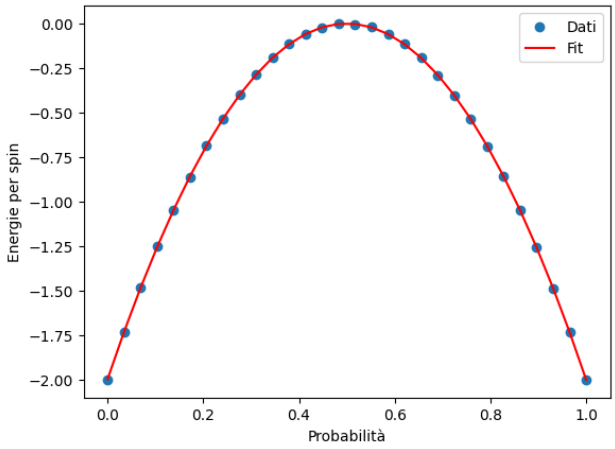


Figure 1: Fit of E/N as a function of p

For code development, however, it is necessary to determine the inverse function, in such a way as to determine the dependence of p on E/N. Hypothesizing also in this case a quadratic dependence, and determining two distinct codomains ([0, 0.5], [0.5, 1]) for bijectivity, the following functions were found:

$$p = 0.101E_N^2 + 0.412E_N + 0.444 \quad , \quad p \in [0, 0.5] \tag{2}$$

$$p = -0.101E_N^2 - 0.412E_N + 0.555 \quad , \quad p \in [0.5, 1] \tag{3}$$

The first function generates predominantly positive magnetization, the second predominantly negative magnetization.

All fits performed did not show explicit dependence on lattice size.

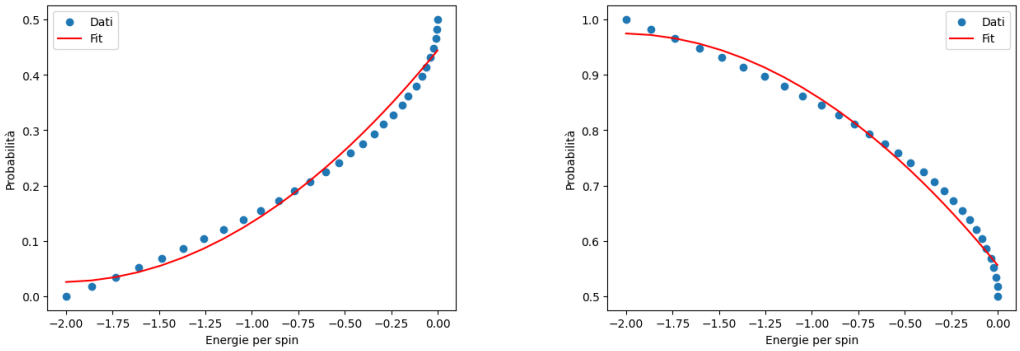


Figure 2: Fit of p as a function of E/N

3.2 Dependence of Mean Magnetization per Spin on Energy per Spin

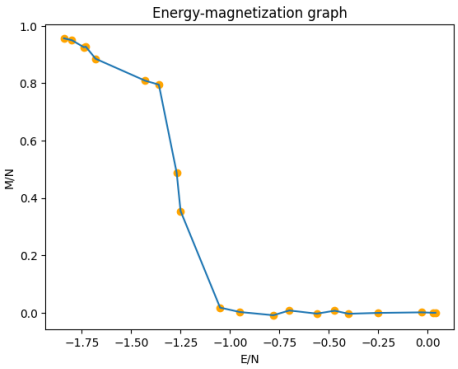
As indicated in the problem text, the mean magnetization per spin as a function of energy per spin was calculated, generating 20 data pairs. These were obtained from the `Model.driver()` subroutine, which works by default first creating a list of equispaced energy points in the interval $[E_0, 0]$, where

$$E_0 = -2\left(1 - \frac{1}{L}\right) \tag{4}$$

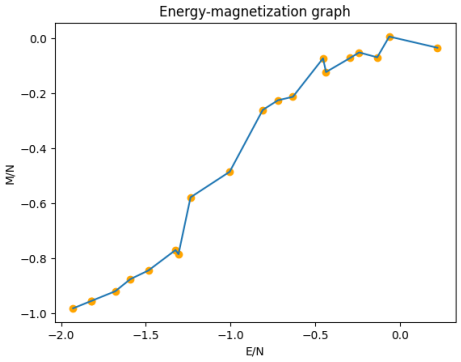
is the minimum energy per spin that a square lattice of side L can reach given the periodic boundary conditions. Then, the probabilities necessary to obtain such energies are calculated through the fit functions obtained previously. Finally, the spin lattices are initialized with such probabilities and evolved for 5000 time steps, after 100 initial steps necessary to equilibrate the magnetizations.

The default equilibration steps for lattices were determined by evaluating the magnetizations of lattices of various sizes after different numbers of time steps. It was concluded that 100 steps are sufficient for lattices of even very large dimensions, as can be observed graphically from the attached .gif file.

The graph with positive magnetization in Figure 3 suggests rather strongly that there is a phase transition around the value of E/N of -1.25. The graph with negative magnetization seems to present, instead, a more homogeneous dependence of M/N on E/N, almost linear. Using other seeds for the random number generator, one can see how this is a particular case, and also in the case of negative magnetizations there seems to be a phase transition, although less marked.



(a) Positive magnetization



(b) Negative magnetization

Figure 3: Energy-magnetization graphs

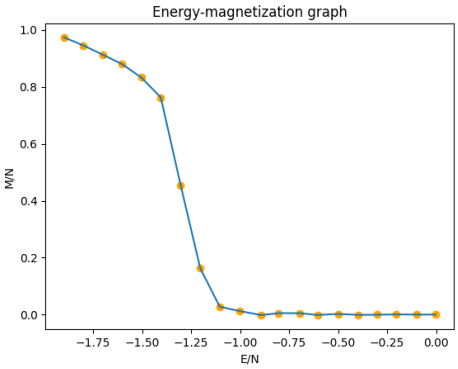
3.3 Generation of Lattices with Exact Energy

To improve the graphs, one possibility is to evolve several lattices at the same energy E/N and subsequently average the various magnetizations per spin. This approach implicitly involves a problem: initializing lattices at an exact energy E_N^*. One solution is to initialize a lattice randomly at an initial energy per spin E/N close to E_N^*, and then flip some spins in such a way as to increase or decrease E/N.

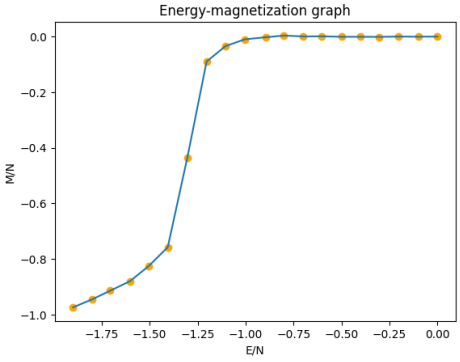
The subroutine responsible for approaching a given energy is structured as follows: in a loop, the difference $E_N^* - E/N$ is calculated. If positive, an attempt is made to increase E/N; if negative, to decrease it. To increase it, a spin is randomly identified, the energy change dE that flipping it would entail is evaluated, and it is flipped only if dE is positive. The opposite is done to attempt to decrease the energy. In doing so, a potential problem arises that, if not adequately managed, does not lead to any relevant energy change. The latter is a function of spin values that, in general, presents a very high number of local minima and maxima. If between the initial energy per spin and E_N^* there is at least one minimum or maximum, the subroutine will uselessly attempt to increase or decrease the energy per spin, not succeeding since the spin configuration is positioned at the extremum. The problem could be solved with the Metropolis algorithm, but working in this context in the microcanonical ensemble we do not have temperature as a control parameter of the system. Hence the fundamental importance of generating initial energies with appropriate probabilities, in such a way as to initialize a lattice with E/N in a neighborhood of E_N^* devoid of local extrema.

Below are the graphs with the same parameters in the driver used previously, with the difference that each data pair is an average of 15 pairs at

fixed energy. One can observe how the graphs are significantly improved, now making evident a phase transition around $E/N = -1.25$ for both negative and positive magnetization.



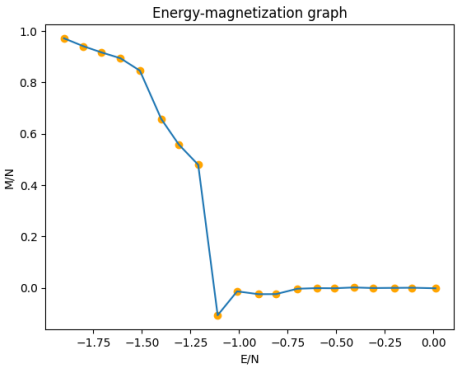
(a) Positive magnetization



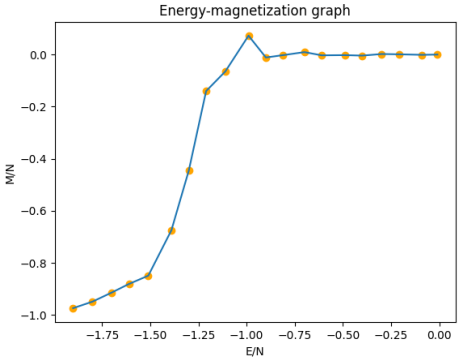
(b) Negative magnetization

Figure 4: 15 iterations for each E/N

For completeness, Figure 5 shows graphs of some cycles where only the `Spin_matrix.adjust_energy()` subroutine is used, without averaging over multiple values of E/N.



(a) Positive magnetization



(b) Negative magnetization

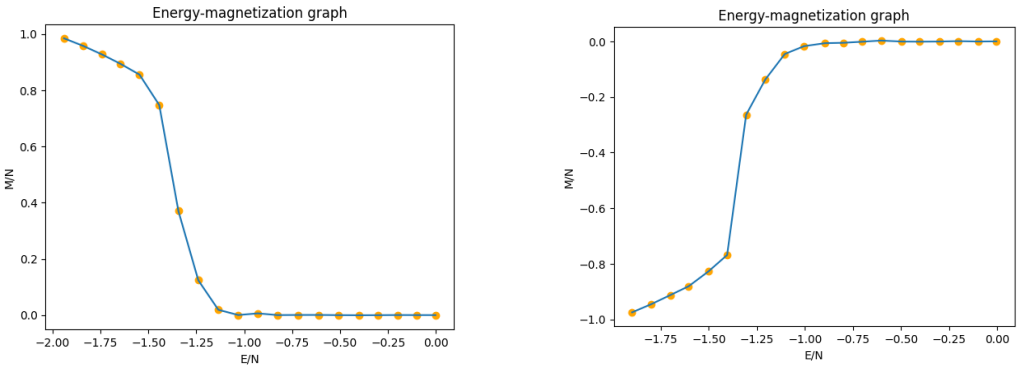
Figure 5: Use of the `Spin_matrix.adjust_energy()` subroutine

One can therefore observe that the mere generation of data with homogeneously distributed energies improves the results obtained.

4 Point (d)

In the configuration space of a lattice, two sets of configurations at the same energy can be disjoint. To pass from one set to another, it is necessary for the system dynamics to explore configurations at different energies. During temporal evolution given by cellular automata, the energy of a lattice is fixed. One can therefore conclude that such dynamics is not ergodic.

The doubt may emerge that the data pairs obtained previously, each belonging to its own set of lattice configurations, cannot be considered representative of the dynamics. To overcome this problem, a routine was written that allows periodically changing a random spin within the lattice, in such a way as to maintain the system energy within a determined interval during temporal evolution. Figure 6 shows graphs of a driver execution where lattices are allowed to change energy within an interval of 1/10 of the minimum energy E_0 reachable by the lattice.



(a) Positive magnetization

(b) Negative magnetization

Figure 6: Ergodic dynamics

The graphs do not change significantly from the non-ergodic case, thus indicating that the random choice of configurations for a given energy provides a representative dependence of mean magnetization per spin M/N on energy per spin E/N.

5 Point (b)

Figure 7 shows the graph of various executions of the driver with positive magnetization with the same parameters as point (d), for different values of lattice dimensions. L indicates the number of spins per side of the lattice. Theoretically, we know that in the canonical ensemble of an infinite-

dimensional lattice there is a phase transition of magnetization around a critical temperature T_C. To this temperature corresponds in the microcanonical ensemble a critical energy E_C. From the last graph, one can estimate that with increasing L the system phase transition approaches this value asymptotically, since the curves not only slightly increase their slope around their own critical value but also shift slightly to the left.

One can therefore conclude that periodic boundary conditions affect the simulation, although already for a lattice of side 20, like the one used in the previous discussion, the critical energy does not deviate excessively from the true value.

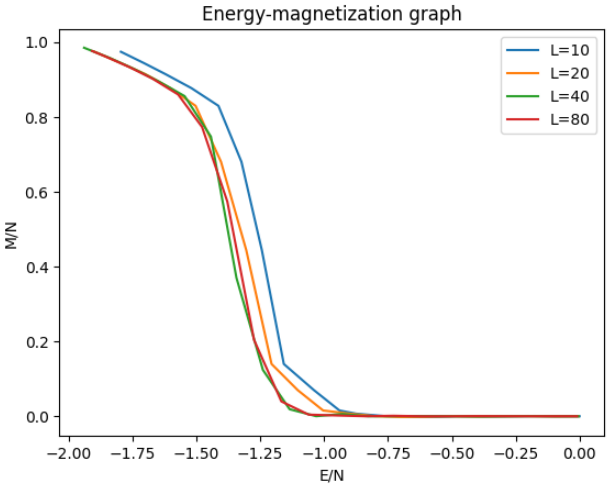


Figure 7: Energy-magnetization graphs for different lattice sizes