Exercise 01: Simulation of the 1-D Ising model

Computational Physics WS20/21

Marc Vöhringer Carrera und Helen Meyer

November 11, 2020

1 Introduction

The Ising-Model describes a lattice in which there are N particles at sites labeled by $x \in \{1, ..., N\}$, which have two possible spin states. The possible spins $s_x \in \{\pm 1\}$ are opposed to each other. In this exercise we want to implement the Ising-Model in one dimension, meaning we consider a chain with N spins. The different possibilities for a specific spin state are called spin configurations $\mathbf{s} = \{s_x : \forall x\}$.
The Hamiltonian

$$\mathcal{H}(\mathbf{s}) = -J \sum_{\langle x,y \rangle} s_x s_y - h \sum_x s_x \tag{1}$$

defines the dynamic of this system when immersed in a heat bath of temperature T in an external magnetic field h. Here $\langle x, y \rangle$ is a nearest-neighbour pair of spins x and y, J is a real number.

1: The parameter J defines the coupling of two spins (in this case we assume that J is unequal to zero only for neighbouring spins). For a postive J the energy of the system is decreased when neighbouring spins are parallel to each other, and increased otherwise. This coupling is called "ferromagnetic" and is the reason for the formation of magnetic domains in ferromagnets. For a negative J there is a decrease in energy if neighbouring spins are anti-parallel, which corresponds to "anti-ferromagnetic" coupling.

2: We also assume periodic boundary conditions, i. e., the nearest neighbour to x_N would be x_1. Imagining the one dimensional lattice of length N as a string, periodic boundary conditions would mean attaching the ends of the string to each other. This implies that spin configurations which can be transformed into another by rotation of the spin sites by j lattice sites

$$x_i \to x_{i+j} \quad \text{where} \quad x_k = x_{k \bmod N} \quad \text{for} \quad k > N, \tag{2}$$

yield the same energy. The propability for finding a spin system at configuration $\mathbf{s}$ is given by the Boltzmann distribution

$$P(\mathbf{s}) = \exp\left(-\frac{\mathcal{H}(\mathbf{s})}{k_B T}\right) / \sum_{\mathbf{s}\prime} \exp\left(-\frac{\mathcal{H}(\mathbf{s}\prime)}{k_B T}\right) \equiv \frac{1}{Z} \exp\left(-\frac{\mathcal{H}(\mathbf{s})}{k_B T}\right), \tag{3}$$

where

$$Z = \sum_{\mathbf{s}\prime} \exp\left(-\frac{\mathcal{H}(\mathbf{s}\prime)}{k_B T}\right) \tag{4}$$

is the partition function. It is summed over all possible spin configurations.

3: For simplicity we work with units where $k_B = 1$. The probability $P(\mathbf{s})$ is therefore given by

$$P(\mathbf{s}) = \frac{1}{Z} \exp\left(\frac{J}{T} \sum_{\langle x,y \rangle} s_x s_y + \frac{h}{T} \sum_x s_x\right), \tag{5}$$

showing that the relevant ratios for this problem are J/T and h/T. Specifically, this means that the solution for the problem only depends on these two ratios. The result does not change, if the Temperature T, as well as the neighbouring coupling J and external field coupling h, are varied in the same fashion. For example increasing T, J and h all by a factor of 2, yields the same result. From now on, if we just write h or J, we always mean the ratios h/T and J/T.

2 Deterministic implementation of the Ising-Model

Our first approach was to solve the problem deterministically by generating all possible spin configurations for a lattice of size N. The procedure for this is described in the code, which is contained in the file `ising_deterministic.py`. The result for the average magnetization $\langle m \rangle$ can then just be obtained by averaging it over all possible spin configurations and over all spin sites. The code calculates the average magnetization for all possible combinations of $h \in [-1, 1]$ in steps of 0.01, and $N \in [2, 20]$. The results obtained by running `ising_deterministic.py` are very good and close to the analytical result as can be seen when comparing Fig. (2.0.1) and (2.0.2) to Fig. (4.0.1).

However, even for $N \leq 20$ this code needs several hours to run (also because the stepsize in h is pretty small, but even for smaller stepsizes calculating the average magnetization for one combination of h and J takes a few seconds). This algorithm is inefficient, as the computation time increases exponentially in N, because the number of possible spin combinations that have to be generated, is 2^N. We therefore are looking for a way to make our algorithm faster, such that the runtime does not increase exponentially in N.

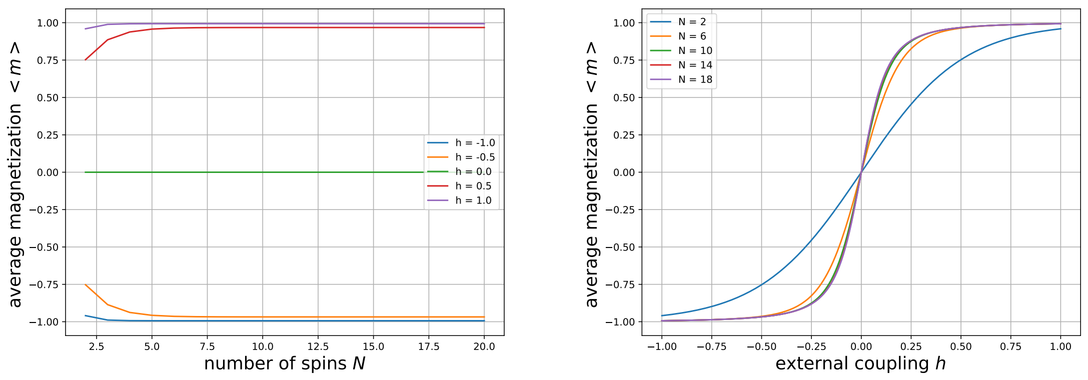


Figure 2.0.1: Average magnetization $\langle m \rangle$ depending on number of sites N for certain magnetic coupling h (left) and depending on h for certain N (right).

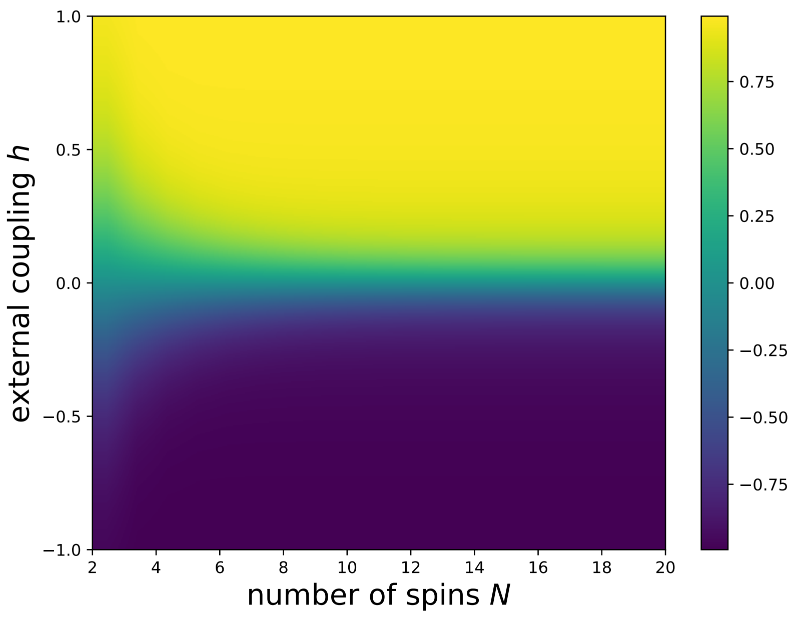


Figure 2.0.2: 2D plot showing the dependence average magnetization $\langle m \rangle$ on N and h.

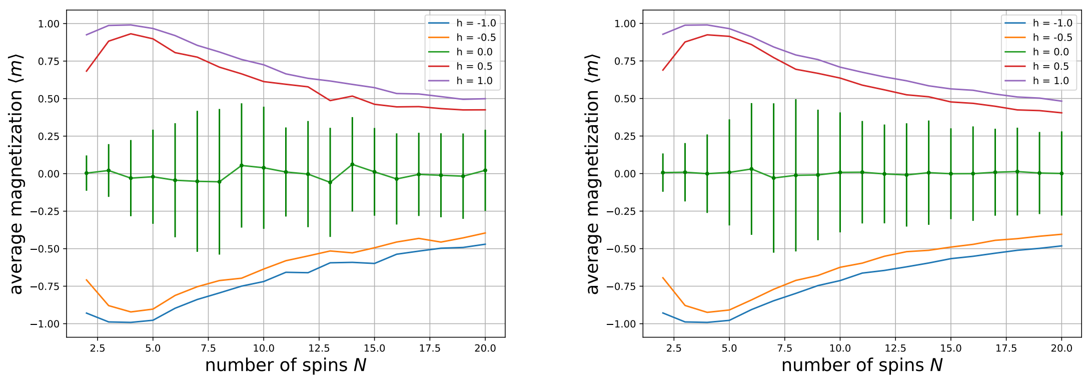


Figure 3.0.1: Average magnetization $\langle m \rangle$ depending on N for certain h. Left: For $N_{\text{meas}} = 100$, $h_{\text{step}} = 0.1$, Right: $N_{\text{meas}} = 1000$, $h_{\text{step}} = 0.1$.

3 Random sampling of spin configurations

To make the algorithm faster for large N, we want to sample the spin configurations, by choosing between $s_i \in \{-1, 1\}$ randomly for all N lattice sites. By doing this n times, we get n random spin configurations. The idea behind the algorithm is, that for $n \to \infty$, $\langle m \rangle$ converges against the true value. As we can not increase n arbitrarily, because the simulation would then take arbitrarily long, we choose a fixed n, and repeat this process N_{meas} times. That way, by averaging over all N_{meas} "measurements", we get for the average magnetization

$$\langle m \rangle = \sum_{i=0}^{N_{\text{meas}}-1} \langle m \rangle_i, \tag{6}$$

where $\langle m \rangle_i$ is the average magnetization for a single measurement, a result which should be close to the true magnetization. By taking the standard deviation of the $\{\langle m \rangle_i\}$, we get an estimate for the error $\Delta \langle m \rangle$ of the average magnetization $\langle m \rangle$. As we will see, even though this approach sounds reasonable, we will see later that it fails. The reason for that will be discussed after the comparison with the analytic solution. We run the code `ising_sum.py` three times for $n = 100$ configurations: first we choose $N_{\text{meas}} = 100$, secondly $N_{\text{meas}} = 1000$ both for steps in h of $h_{\text{step}} = 0.1$. Finally, we run the code for $N_{\text{meas}} = 100$ and smaller steps (0.01) for h. The results are displayed in Fig. (3.0.1) - (3.0.3). In the next section we want to compare these to the analytic solution.

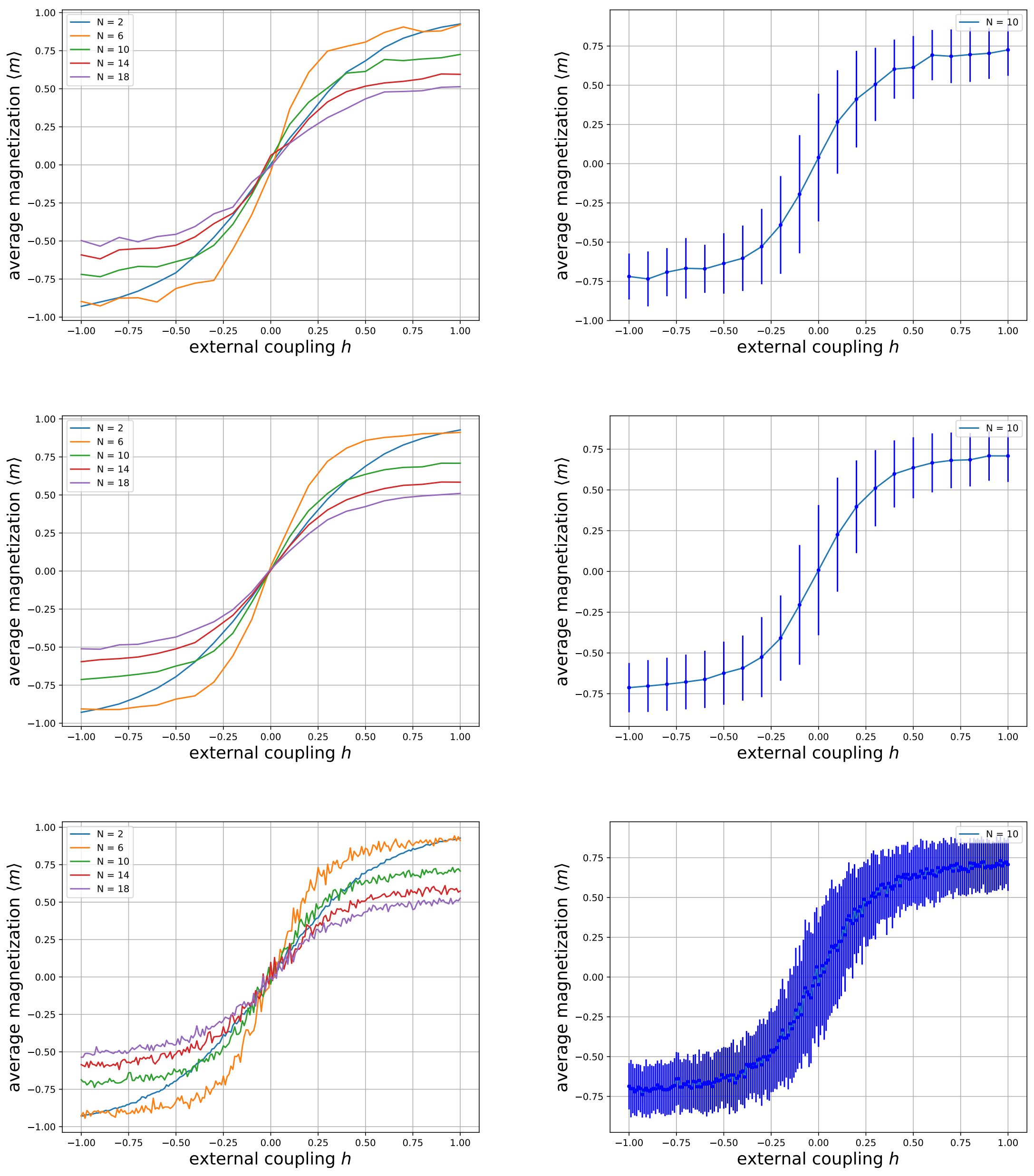


Figure 3.0.2: Average magnetization $\langle m \rangle$ depending on h for certain N. Top: For $N_{\text{meas}} = 100$, $h_{\text{step}} = 0.1$, Middle: $N_{\text{meas}} = 1000$, $h_{\text{step}} = 0.1$, Bottom: $N_{\text{meas}} = 100$, $h_{\text{step}} = 0.01$.

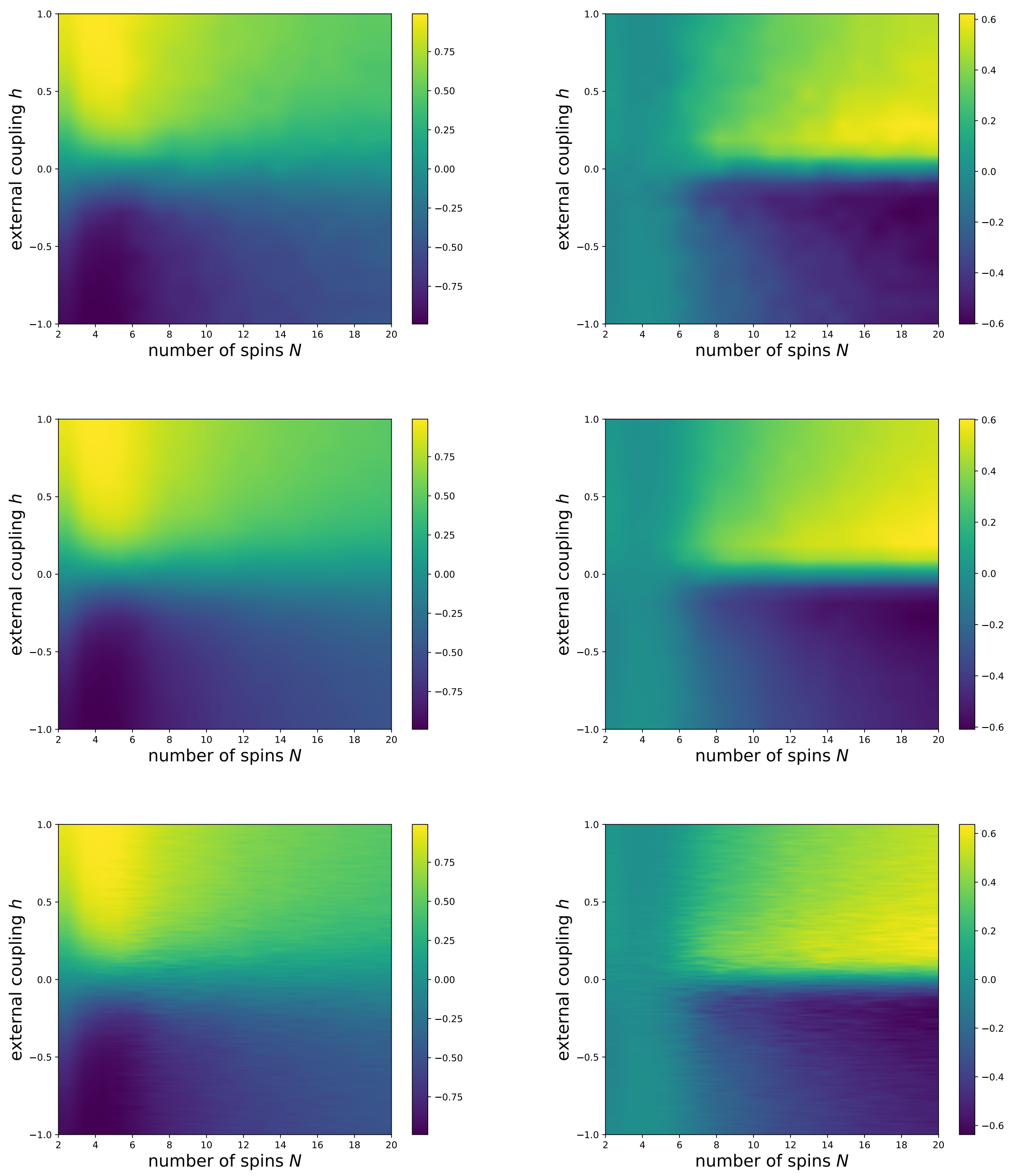


Figure 3.0.3: Average magnetization $\langle m \rangle$ (left) and the absolute deviation from the analytical result δ (right) displayed in a 2D-plot depending on h and N. Top: For $N_{\text{meas}} = 100$, $h_{\text{step}} = 0.1$, Middle: $N_{\text{meas}} = 1000$, $h_{\text{step}} = 0.1$, Bottom: $N_{\text{meas}} = 100$, $h_{\text{step}} = 0.01$.

4 Comparison to analytic Solution

For the Ising-Modell in one dimension Z can be calculated analytically:

$$Z = \lambda_+^N + \lambda_-^N; \qquad \lambda_\pm = e^{\frac{J}{T}} \left(\cosh \frac{h}{T} \pm \sqrt{\sinh \frac{h}{T}^2 + e^{-4\frac{J}{T}}} \right). \tag{7}$$

By obtaining an analytic solution of Z according to Eqn. 7, we can calculate the average magnetization by taking the derivative of the logarithm of Z as follows:

$$\langle m \rangle = \frac{T}{N} \frac{\partial \log Z}{\partial h}. \tag{8}$$

The analytic solution for $\langle m \rangle$ can thus be obtained in the code by calculating

$$\langle m \rangle = \frac{T}{NZ} \left(\frac{\partial \lambda_+^N}{\partial h} + \frac{\partial \lambda_-^N}{\partial h} \right), \tag{9}$$

where

$$\frac{\partial \lambda_\pm^N}{\partial h} = \frac{N}{T} \cdot \lambda_\pm^{N-1} \cdot e^{\frac{J}{T}} \left(\sinh \frac{h}{T} \pm \frac{\sinh \frac{h}{T} \cosh \frac{h}{T}}{\sqrt{\sinh \frac{h}{T}^2 + e^{-4\frac{J}{T}}}} \right) \tag{10}$$

Using Eqn. 9, we compute the analytic solution for $\langle m \rangle$ together our numeric solution in the script `ising_sum.py`. The results for the analytic solution can be seen in Fig. (4.0.1).

As can be seen from the plots by comparing the numeric to the analytic solutions, the numeric algorithm using random spin configuration samples, fails for large values of N in the cases of $|h| > 0.2$, which is very obvious when looking at the absolute deviation in Fig. (3.0.3). Of course the more measurements we take ($N_{\text{meas}} = 1000$), the better the results obtained by random sampling, but the longer the code takes.

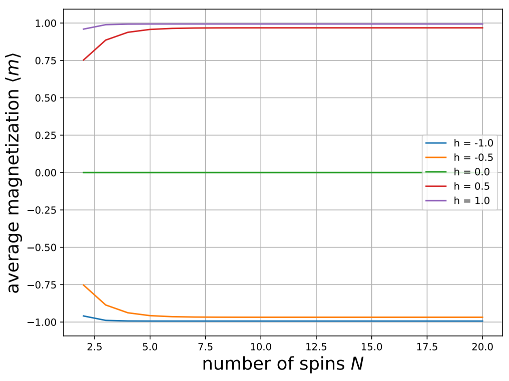
5 The importance of importance sampling

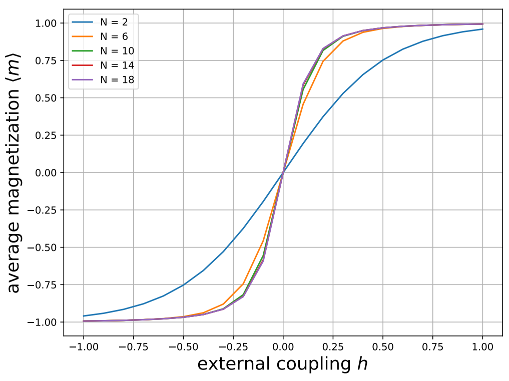
The reason for the uniform sampling algorithm failing is, that as in the cases where the true value of $\langle m \rangle \approx \pm 1$, the only spin configurations significantly contributing to the results, are the ones where all spins point upwards or downwards. The probability of generating these spin configurations randomly is

$$P\left(\langle m \rangle = 1\right) = \frac{1}{2^N}. \tag{11}$$

This means that already for $N = 10$ the probability of generating this configuration randomly in one measurement run using $n = 100$ configurations is

$$P_{\text{single\_meas}} = 1 - (1 - \frac{1}{2^{10}})^{100} \approx 0,003, \tag{12}$$





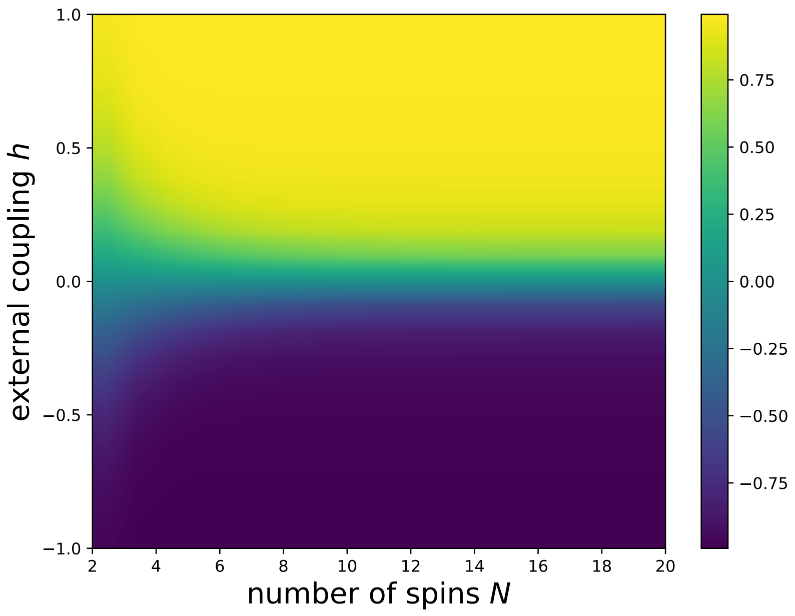


Figure 4.0.1: Analytical results for the magnetization $\langle m \rangle$.

which means that the probability of **not** generating it during $N_{\text{meas}} = 100$ runs is

$$P_{\text{100\_meas}} = (1 - P_{\text{single\_meas}})^{100} \approx 0,74. \tag{13}$$

For $N_{\text{meas}} = 1000$ the stakes look better, but even if the configuration is considered in two or three runs, as all the individual results are weighted equally for the mean value, the estimate of $\langle m \rangle$ will still be way off. For that reason it is necessary to implement importance sampling.